User Manual

Allen-Bradley

Motion Coordinate System

Catalog Numbers 1756-HYD02, 1756-M02AE, 1756-M02AS, 1756-M03SE, 1756-M08SE, 1756-M16SE, 1768-M04SE

Allen-Bradley • Rockwell Software

Rockwell Automation

Important User Information

Read this document and the documents listed in the additional resources section about installation, configuration, and operation of this equipment before you install, configure, operate, or maintain this product. Users are required to familiarize themselves with installation and wiring instructions in addition to requirements of all applicable codes, laws, and standards.

Activities including installation, adjustments, putting into service, use, assembly, disassembly, and maintenance are required to be carried out by suitably trained personnel in accordance with applicable code of practice.

If this equipment is used in a manner not specified by the manufacturer, the protection provided by the equipment may be impaired.

In no event will Rockwell Automation, Inc. be responsible or liable for indirect or consequential damages resulting from the use or application of this equipment.

The examples and diagrams in this manual are included solely for illustrative purposes. Because of the many variables and requirements associated with any particular installation, Rockwell Automation, Inc. cannot assume responsibility or liability for actual use based on the examples and diagrams.

No patent liability is assumed by Rockwell Automation, Inc. with respect to use of information, circuits, equipment, or software described in this manual.

Reproduction of the contents of this manual, in whole or in part, without written permission of Rockwell Automation, Inc., is prohibited.

Throughout this manual, when necessary, we use notes to make you aware of safety considerations.

	WARNING: Identifies information about practices or circumstances that can cause an explosion in a hazardous environment, which may lead to personal injury or death, property damage, or economic loss.
	ATTENTION: Identifies information about practices or circumstances that can lead to personal injury or death, property damage, or economic loss. Attentions help you identify a hazard, avoid a hazard, and recognize the consequence.
IMPORTANT	Identifies information that is critical for successful application and understanding of the product.

Labels may also be on or inside the equipment to provide specific precautions.

	SHOCK HAZARD: Labels may be on or inside the equipment, for example, a drive or motor, to alert people that dangerous voltage may be present.
	BURN HAZARD: Labels may be on or inside the equipment, for example, a drive or motor, to alert people that surfaces may reach dangerous temperatures.
	ARC FLASH HAZARD: Labels may be on or inside the equipment, for example, a motor control center, to alert people to potential Arc Flash. Arc Flash will cause severe injury or death. Wear proper Personal Protective Equipment (PPE). Follow ALL Regulatory requirements for safe work practices and for Personal Protective Equipment (PPE).

Allen-Bradley, CompactLogix, ControlLogix, Logix5000, RSLogix 5000, SoftLogix, Studio 5000 Logix Designer, and Rockwell Automation are trademarks of Rockwell Automation, Inc.

Trademarks not belonging to Rockwell Automation are property of their respective companies.

This manual contains new and updated information. Changes throughout this revision are marked by change bars as shown to the right of this paragraph.

New and Updated Information

This table contains the changes made to this revision.

Topic	Page
This manual has been reorganized since the last revision. Content from Chapter 3, Cartesian Coordinate System Examples, has been integrated into Appendix A, which describes all of the instructions in detail. Information from chapters 10, 11, 12, and Appendix B has been consolidated into earlier chapters, but no information was deleted.	
Updated the New Tag Dialog Box graphic.	16
Added a list of termination types in Blended Moves and Termination Types.	34
Updated the restriction on MCT and MCTP instructions to list only SoftLogix controllers.	109
Updated field descriptions in Operands - Relay Ladder table as marked.	111
Updated field descriptions in MCCM Instruction Operands - Relay Ladder table as marked.	138
Updated field descriptions in MCCD Instruction Operands - Relay Ladder table as marked.	187
Updated the Guidelines for Programming an MCT instruction and when to start motion.	211
Added descriptions for the EN, DN, ER, PC, IP, and AC bits to the table describing Status Bits for Motion Instructions (MCLM, MCCM) when MDCC Is Active.	234
Updated the description of Error Code 63.	263
Updated the description for error code 75.	265

Notes:

Preface

Studio 5000 Environment 11
Where to Find Sample Projects 12
For More Information 13

Chapter 1

Create and Configure a Coordinate System

Create a Coordinate System 16
Coordinate System Wizard Dialog Boxes 19
Edit Coordinate System Properties 20
- General Tab 21
- Geometry Tab 23
- Units Tab 24
- Offsets Tab 26
- Joints Tab 27
- Dynamics Tab 28
- Dynamics Tab Manual Adjust 30
- Motion Planner Tab 31
- Tag Tab 32

Chapter 2

Cartesian Coordinate System

Program an MCLM Instruction 34
Blended Moves and Termination Types 34
- Example Ladder Diagram for Blended Instructions 35

Bit State Diagrams for Blended Moves 37
- Bit States at Transition Points of Blended Move by Using Actual Tolerance or No Settle 37
- Bit States at Transition Points of Blended Move by Using No Decel 38
- Bit States at Transition Points of Blended Move by Using Command Tolerance 39
- Bit States at Transition Points of Blended Move by Using Follow Contour Velocity Constrained or Unconstrained 40

Choose a Termination Type 41
- Velocity Profiles for Collinear Moves 43
- Symmetric Profiles 45
- Triangular Velocity Profile 47
- Blending Moves at Different Speeds 48

MCLM and MCCM Examples 48

Chapter 3

Kinematics Coordinate Systems

Motion Calculate Transform Position (MCTP) 49
Motion Coordinated Shutdown Reset (MCSR) 49
Useful Terms 50
Gather Information about Your Robot 50
Summary of Kinematic Steps 51

Determine the Coordinate System Type 52

Chapter 4

Articulated Independent Robot

Reference Frame 55
Methods to Establish a Reference Frame 57
Method 1 - Establishing a Reference Frame 58
Method 2 - Establishing a Reference Frame 59
Work Envelope 59
Configuration Parameters 61
Link Lengths 61
Base Offsets 63
End-effector Offsets 64
Delta Robot Geometries 64
Configure a Delta Three-dimensional Robot 65
Establish the Reference Frame for a Delta Three-dimensional Robot 66
Calibrate a Delta Three-dimensional Robot 66
Alternate Method for Calibrating a Delta Three-dimensional Robot 67
Configure Zero Angle Orientations for Delta Three-dimensional Robot 67
Identify the Work Envelope for a Delta Three-dimensional Robot 69
Define Configuration Parameters for a Delta Three-dimensional Robot 71
Configure a Delta Two-dimensional Robot 73
Establish the Reference Frame for a Delta Two-dimensional Robot 74
Calibrate a Delta Two-dimensional Robot 75
Identify the Work Envelope for a Delta Two-dimensional Robot 75
Define Configuration Parameters for a Delta Two-dimensional Robot 76
Configure a SCARA Delta Robot 78
Establish the Reference Frame for a SCARA Delta Robot 78
Calibrate a SCARA Delta Robot 79
Identify the Work Envelope for a SCARA Delta Robot 80
Define Configuration Parameters for a SCARA Delta Robot 80
Configure a Delta Robot with a Negative X1b Offset 82
Arm Solutions 83
Left-Arm and Right-Arm Solutions for Two-Axes Robots 83
Solution Mirroring for Three-dimensional Robots 83
Activating Kinematics 84
Change the Robot Arm Solution 85
Plan for Singularity 85

Encounter a No-solution Position ... 86
Configure a SCARA Independent Robot ... 86
Establish the Reference Frame for a SCARA Independent Robot ... 86
Identify the Work Envelope for a SCARA Independent Robot ... 88
Define Configuration Parameters for a SCARA Independent Robot ... 89
Error Conditions ... 90
Monitor Status Bits for Kinematics ... 91

Chapter 5

Articulated Dependent Robot

Reference Frame ... 93
Methods to Establish a Reference Frame ... 96
Method 1 - Establishing a Reference Frame ... 96
Method 2 - Establishing a Reference Frame ... 97
Work Envelope ... 98
Configuration Parameters ... 99
Link Lengths ... 99
Base Offsets ... 100
End-effector Offsets ... 101

Chapter 6

Configure a Cartesian Gantry Robot

Establish the Reference Frame for a Cartesian Gantry Robot ... 103
Identify the Work Envelope for a Cartesian Gantry Robot ... 104
Define Configuration Parameters for a Cartesian Gantry Robot ... 104

Chapter 7

Configure a Cartesian H-bot

About Cartesian H-bots ... 105
Establish the Reference Frame for a Cartesian H-bot ... 106
Identify the Work Envelope for a Cartesian H-bot ... 106
Define Configuration Parameters for a Cartesian H-bot ... 107

Appendix A

Motion Coordinated Instructions (MCLM, MCCM, MCCD, MCS, MCSD, MCT, MCTP, MCSR, MDCC)

Motion Coordinated Linear Move (MCLM) ... 109
Operands ... 110
Motion Control Bits ... 115
Move Type ... 116
Velocity Profiles ... 122
Convert Jerk Units ... 125
Merge ... 126
Coordinated Motion Merge Example ... 126
Programming Guidelines for Zero Length Moves ... 130

MCLM Target Position Entry Dialog 131
Arithmetic Status Flags 132
Fault Conditions 133
Error Codes 133
MCLM Changes to Status Bits 134
Motion Coordinated Circular Move (MCCM) 136
Operands 137
Motion Control Bits 144
Move Type 144
Via/Center/Radius 145
Two-Dimensional Arc and Circle Examples 146
MCCM with Rotary Axes Examples 161
Three-dimensional Arc Examples 166
Calculate Jerk Units 170
Merge 172
Programming Guidelines for Zero Length Moves 172
MCCM Target Position Entry Dialog Box 174
Arithmetic Status Flags 176
Fault Conditions 176
Error Codes 177
Circular Error Examples 178
Circular Programming Reference Guide 183
MCCM Changes to Status Bits 183
Master Driven Speed Control (MDSC) and Motion Direct Command Support 185
Motion Coordinated Change Dynamics (MCCD) 185
Operands 186
Motion Type 190
Impact of Changes to Acceleration and Deceleration Values on Motion Profile 191
Arithmetic Status Flags 192
Fault Conditions 192
Error Codes 192
MCCD Changes to Status Bits 193
Motion Coordinated Stop (MCS) 194
Operands 194
Motion Control Bits 196
How Stop Types Affect Transforms 197
Master Driven Speed Control (MDSC) and the MCS Instruction 200
Arithmetic Status Flags 200
Fault Conditions 201
Error Codes 201
MCS Changes to Status Bits 201
Motion Coordinated Shutdown (MCSD) 202
Operands 202

Motion Control Bits 203
Master Driven Speed Control (MDSC) and the MCSD Instruction 203
Arithmetic Status Flags 203
Fault Conditions 203
Error Codes 204
MCSD Changes to Status Bits 204
Motion Coordinated Transform (MCT) 205
Operands 205
Motion Control Bits 207
Data Flow of MCT Instruction Between Two Coordinate Systems 207
Programming Guidelines 209
Arithmetic Status Flags 211
Fault Conditions 211
Error Codes 211
MCT Changes to Status Bits 212
Example 1 - Pick and Place Ladder Diagram 213
Example 2 - Pick and Place - Structured Text 214
Example 3 - Change Orientation 215
Example 4 - Change Translation 216
Motion Calculate Transform Position (MCTP) 217
Operands 217
Example: Enter a transform direction of Inverse Left Arm as InverseLeftArm 219
Programming Guidelines 220
Data Flow of MCTP Instruction Between Two Coordinate Systems 221
Arithmetic Status Flags 222
Fault Conditions 222
Error Codes 222
MCTP Changes to Status Bits 223
Example 1 - Calculate Position 223
Example 2 - Change Orientation 225
Example 3- Change Translation 225
Example 4 - Change Direction 226
Motion Coordinated Shutdown Reset (MCSR) 227
Operands 227
Motion Control Bits 228
Arithmetic Status Flags 228
Fault Conditions 228
Error Codes 228
MCSR Changes to Status Bits 228
Structured Text 229
Master Driven Coordinate Control (MDCC) 229
Operands 230

Motion Direct Command and the MDCC Instruction 231
MOTION_INSTRUCTION Bit Leg Definitions for MDCC 232
Arithmetic Status Flags 232
Fault Conditions for Motion Instructions when MDCC Is Active 233
Error Codes 234
Status Bits for Motion Instructions (MCLM, MCCM) when MDCC Is Active 234
Coordinated Motion Status Bits 238
Changing Between Master Driven and Time Driven Modes for Coordinated Motion Instructions 238
Changing the Master Axis 239
Input and Output Parameters Structure for Coordinate System Motion Instructions 240
Speed, Acceleration, Deceleration, and Jerk Enumerations for Coordinated Motion 250
Speed Enumerations 250
Acceleration and Deceleration Enumerations 250
Jerk Enumerations 252

Appendix B

Coordinate System Attributes

How to Access Attributes 255
Coordinate System Attributes 255

Appendix C

Error Codes (ERR) for Coordinate Motion Instructions

Additional Error Code Information 267

AppendixD

History of Changes

MOTION-UM002C-EN-P, September 2012 269
MOTION-UM002B-EN-P, November 2011 269
MOTION-UM002A-EN-P, January 2010 270

Index

. 271

This manual provides information on configuring a variety of coordinated motion applications. Appendix A provides detailed information about the coordinated motion instructions. Refer to the Additional Resources for information configuration and startup of Sercos and analog motion or Integrated Motion on EtherNet/IP networks.

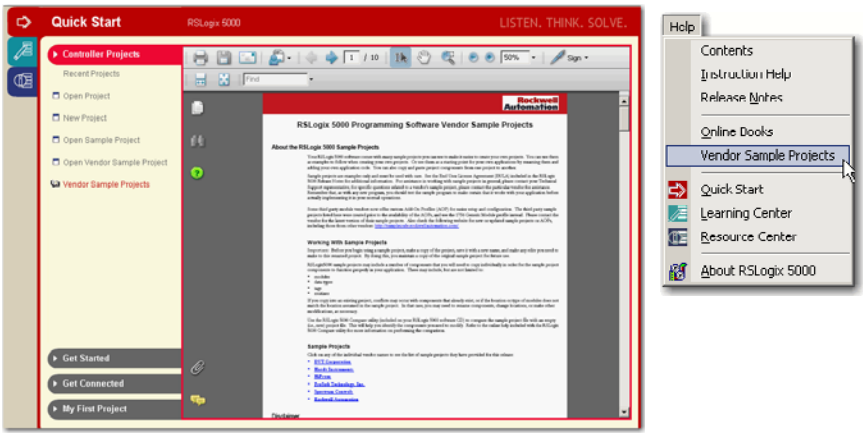
Studio 5000 Environment

The Studio 5000 Automation Engineering & Design Environment™ combines engineering and design elements into a common environment. The first element is the Studio 5000 Logix Designer™ application. The Logix Designer application is the rebranding of RSLogix™ 5000 software and will continue to be the product to program Logix5000™ controllers for discrete, process, batch, motion, safety, and drive-based solutions.

The Studio 5000® environment is the foundation for the future of Rockwell Automation® engineering design tools and capabilities. The Studio 5000 environment is the one place for design engineers to develop all of the elements of their control system.

Where to Find Sample Projects

Use the Logix Designer application Start Page (Alt F9) to find the sample projects.



The Rockwell Automation sample project's default location is:

C:\Users\Public\Documents\Studio 5000\Samples\ENU

There is a PDF file named Vendor Sample Projects on the Start Page that explains how to work with the sample projects. Free sample code is available at: http://samplecode.rockwellautomation.com/.

For More Information

These documents contain additional information concerning related products from Rockwell Automation.

Resource	Description
Sercos and Analog Motion Configuration and Start-up User Manual, publication MOTION-UM001	Describes how to configure a motion application and to start up your motion solution by using Logix5000 motion modules.
Logix5000 Controller Motion Instructions Reference Manual, publication MOTION-RM002	Provides a programmer with details about motion instructions for a Logix-based controller.
Integrated Motion on the Ethernet/IP Network Configuration and Startup User Manual, publication MOTION-UM003	Describes how to configure an integrated motion application and to start up your motion solution by using the Studio 5000 Logix Designer™ application.
Logix5000 Controllers Common Procedures, publication 1756-PM001	Provides detailed and comprehensive information about how to program a Logix5000 controller.
Logix5000 Controllers General Instructions Reference Manual, publication 1756-RM003	Provides a programmer with details about general instructions for a Logix-based controller.
Logix5000 Controllers Process and Drives Instructions Reference Manual, publication 1756-RM006	Provides a programmer with details about process and drives instructions for a Logix-based controller.
ControlLogix Controller User Manual, publication 1756-UM001	Describes the necessary tasks to install, configure, program, and operate a ControlLogix system.
CompactLogix 5730 Controllers User Manual, publication 1769-UM021	Describes the necessary tasks to install, configure, program, and operate a CompactLogix system.
GuardLogix 5570 Controllers User Manual, publication 1756-UM022	Provides information on how to install, configure, program, and use GuardLogix 5570 controllers in Studio 5000 Logix Designer projects.
GuardLogix 5570 Controller Systems Safety Reference Manual, publication 1756-RM099	Contains detailed requirements for achieving and maintaining SIL 3/PLe with the GuardLogix 5570 controller system, using the Studio 5000 Logix Designer application.
Analog Encoder (AE) Servo Module Installation Instructions, publication 1756-IN047	Provides installation instructions for the Analog Encoder (AE) Servo Module, Catalog Number 1756- M02AE.
ControlLogix SERCOS interface Module Installation Instructions, publication 1756-IN572	Provides installation instructions for the ControlLogix SERCOS interface modules, Catalog Number 1756- M03SE, 1756-M08SE, 1756-M16SE, 1756-M08SEG.
CompactLogix SERCOS interface Module Installation Instructions, publication 1768-IN005	Provides installation instructions for the CompactLogix SERCOS interface Module, Catalog Number 1768- M04SE.
Industrial Automation Wiring and Grounding Guidelines, publication 1770-4.1	Provides general guidelines for installing a Rockwell Automation industrial system.
Product Certifications website, http://www.ab.com	Provides declarations of conformity, certificates, and other certification details.

You can view or download publications at http://www.rockwellautomation.com/literature/. To order paper copies of technical documentation, contact your local Allen-Bradley distributor or Rockwell Automation sales representative.

Notes:

Create and Configure a Coordinate System

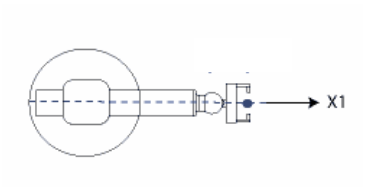
Topic	Page
Create a Coordinate System	16
Configure Coordinate System	17
Coordinate System Wizard Dialog Boxes	19
Edit Coordinate System Properties	20

In Logix Designer application, you use the Coordinate System tag to configure a coordinate system. A coordinate system is a grouping of one or more primary and ancillary axes that you create to generate coordinated motion.

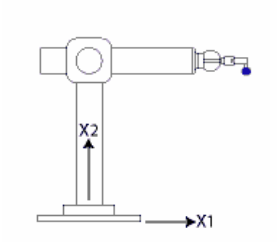
You can configure the coordinate system with one, two, or three dimensions. Logix Designer application supports these types of geometry:

- Cartesian
- Articulated Dependent
- Articulated Independent
- Selective Compliant Assembly Robot Arm (SCARA) Independent
- Delta three-dimensional
- Delta two-dimensional
- SCARA Delta

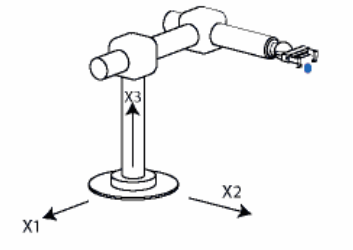
Figure 1 - Coordinate Systems with Orthogonal Axes



Cartesian Coordinate System

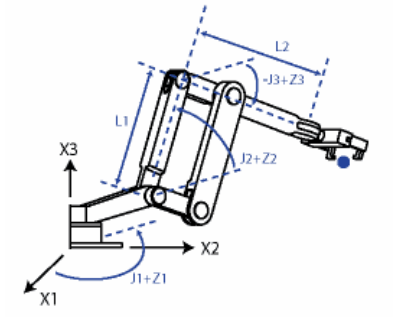


Two-dimensional Cartesian Coordinate System

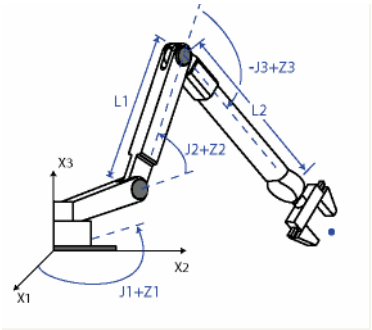


Three-dimensional Cartesian Coordinate System

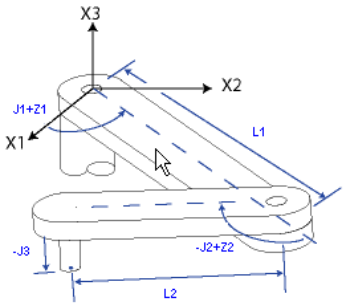
Figure 2 - Coordinate Systems with Non- orthogonal Axes



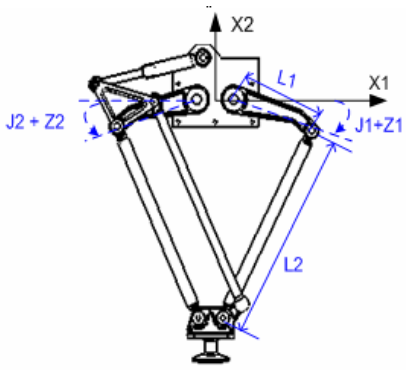
Articulated Dependent Coordinate System



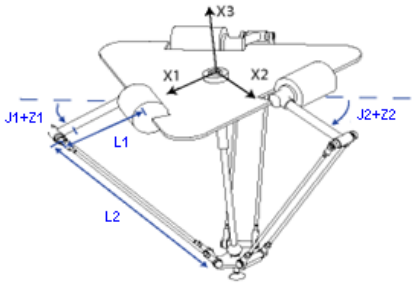
Articulated Independent Coordinate System



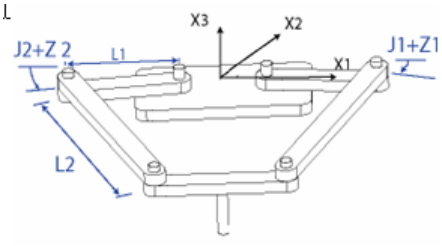
SCARA Independent Coordinate System



Delta Two-dimensional Coordinate System



Delta Three-dimensional Coordinate System



SCARA Delta Coordinate System

Create a Coordinate System

Use the Coordinate System tag to set the attribute values that the Multi-Axis Coordinated Motion instructions use in your motion applications. The Coordinate System tag must exist before you can run any of the Multi-Axis Coordinated Motion instructions.

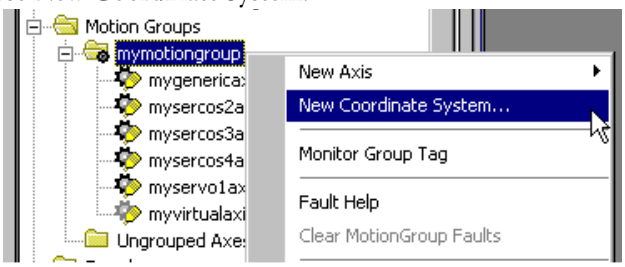
This is where you make the following configurations:

- introduce the COORDINATE_SYSTEM data type,
- associate the coordinate system to a Motion Group,
- associate the axes to the coordinate system,
- set the dimension,
- define the values later used by the operands of the Multi-Axis Motion Instructions.

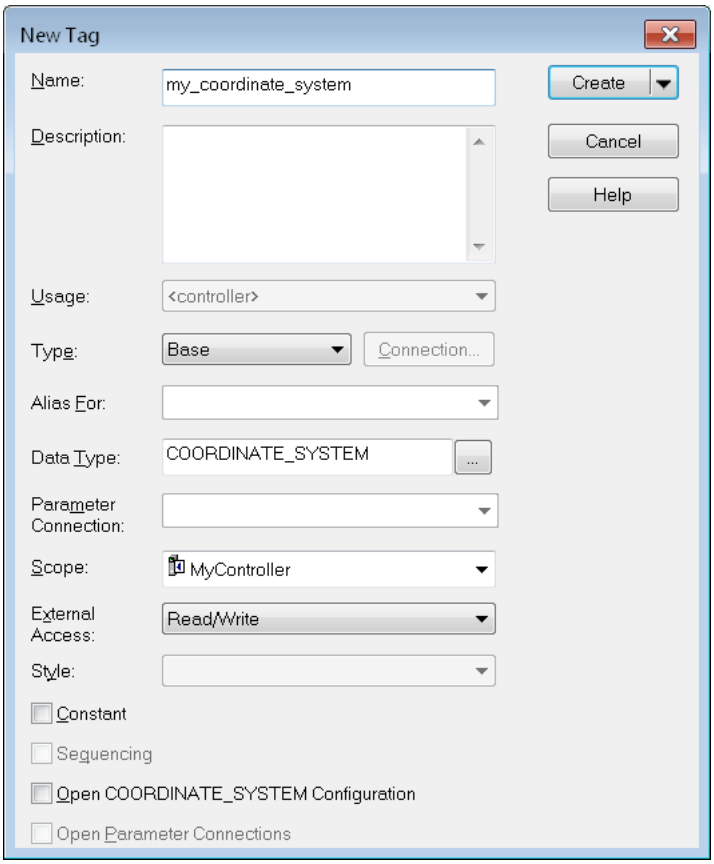
The values for Coordination Units, Maximum Speed, Maximum Acceleration, Maximum Deceleration, Actual Position Tolerance, and Command Position Tolerance are all defined by the information included when the Coordinate System tag is configured.

Follow these steps to create a coordinate system.

1. Right-click the motion group in the Controller Organizer.
2. Choose New Coordinate System.



The New Tag dialog box appears.



Use the parameter descriptions below to help you configure your new tag.

Table 1 - Tag Parameter Descriptions

Parameter Connection	Description
Name	Type a relevant name for the new tag. The name can be up to 40 characters and can be composed of letters, numbers, or underscores (_).
Description	Type a description of the tag. This is an optional field and is used for annotating the tag.
Type	Use the drop-down menu to select what type of tag to create. For a coordinate system, the only valid choices are Base and Alias. Selecting either Produced or Consumed generates an error when the OK button is pressed. • Base refers to a normal tag (selected by default). • Alias refers to a tag that references another tag with the same definition. Special parameters appear on the New Tag dialog box that lets you to identify to which base tag the alias refers.
Alias For	If you selected Alias as the tag Type, enter the name of the associated Base Tag.
Data Type	The Data Type field defines the size and layout of memory that is allocated when the tag is created. Select COORDINATE_SYSTEM.
Scope	Choose the Scope for the tag. The scope defines the range at which tags and routines can be created. A Coordinate System Tag can only be configured at the Controller Scope.
External Access	Choose whether the tag has Read/Write, Read Only, or no (None) access from external applications such as HMIs.
Style	The Style parameter is not activated. No entry for this field is possible. After the information for the tag is entered, you have these options. • Click OK to create the tag and automatically place it in the Ungrouped Axes folder or the Motion Group if the tag was initiated from the Motion Group menu. • Click Open COORDINATE_SYSTEM Configuration to invoke the Coordinate System Tag Wizard after you click Create. The wizard helps you to configure the Coordinate System tag.
Constant	To prevent executing logic from writing values to the tag, check the Constant check box. The state of the Constant check box depends on the type of tag selected. It appears dimmed under the follow conditions. • The tag is an alias tag or a consumed tag. • The FactoryTalk Security action for changing the Constant Value property of a tag is unavailable and the tag is not in the Add-On Instruction definition scope. • You do not have permissions to modify tag properties (the FactoryTalk Security Tag Modified is denied) and that tag is not in the Add-On Instruction definition scope. • The tag's date type is not a Data Table backed type. • The tag's usage is not InOut. • The redundancy controller is in any state that does not allow changes. • The controller has been locked online from another computer. • The controller is safety secured and the tag is a safety tag or a safety mapped tag. • The scope is an equipment phase but the Equipment Phase feature is not activated in the current Logix Designer application license. • The controller is in hard Run mode. • The Add-On Instruction is in Source Protection mode. • You are not allowed to modify Add-On Instructions (FactoryTalk Security Add-On Instruction Modify is Denied) and the tag is in Add-On Instruction definition scope. For details about FactoryTalk Security see FactoryTalk Help: Start > Programs > Rockwell Software > FactoryTalk Tools > FactoryTalk Help. Note: If the properties of the tag modification (for example, Constant Tag property), no longer apply and the Constant check box was previously selected, the Constant check box is not checked.

Click Open COORDINATE_SYSTEM Configuration to display the wizard that guides you through the process of configuring a coordinate system. You can also right-click the tag and choose Properties to access the configuration wizard.

Coordinate System Wizard Dialog Boxes

The Coordinate System Wizard takes you through the Coordinate System Properties dialog boxes. It is not necessary to use the Wizard dialogs to configure your coordinate system. Once it has been created, you can access the Coordinate System Properties dialog box by choosing Properties of the menu. See Edit Coordinate System Properties on page 20 for detailed information about entering configuration information.

Table 2 - Coordinate System Dialog Box Descriptions

Wizard or Dialog Box	Description
General	The General dialog box lets you: • associate the tag to a Motion Group. • enter the coordinate system type. • select the Dimension for the tag (that is, the number of associated axes). • specify the number of dimensions to transform. • enter the associated axis information. • choose whether to update Actual Position values of the coordinate system automatically during operation. This dialog box has the same fields as the General tab found under Coordinate System Properties.
Geometry	The Geometry dialog box lets you configure key attributes related to non-Cartesian geometry and shows the bitmap of the associated geometry.
Offset	The Offset dialog box lets you configure the offsets for the base and end effector. This dialog box shows the bitmaps for the offsets related to the geometry.
Units	The Units dialog box lets you determine the units that define the coordinate system. At this dialog box you define the Coordination Units and the Conversion Ratios. This dialog box has the same fields as the Units tab found under Coordinate System Properties.
Dynamics	Use the Dynamics dialog box for entering the Vector values used for Maximum Speed, Maximum Acceleration, and Maximum Deceleration. It is also used for entering the Actual and Command Position Tolerance values. This dialog box has the same fields as the Dynamics tab found under Coordinate System Properties.
Manual Adjust	The Manual Adjust button is inactive when creating a Coordinate System tag via the Wizard dialog boxes. It is active on the Dynamics tab of the Coordinate System Properties dialog box. It is described in detail in the Editing Coordinate System Properties later in this chapter.
Tag	The Tag dialog box lets you rename your Tag, edit your description, and review the Tag Type, Data Type, and Scope information. The only fields that you can edit on the Tag dialog box are Name and Description. These are the same fields as on the New Tag dialog box and the Coordinate System Properties Tag tab.

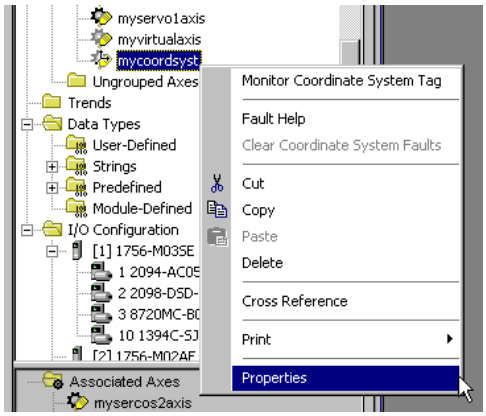
Edit Coordinate System Properties

Create your Coordinate System in the New Tag dialog box, and then configure it. If you did not use the Wizard dialog boxes available from the Create button on the New Tag dialog box, you can make your configuration selections from the Coordinate System Properties dialog box.

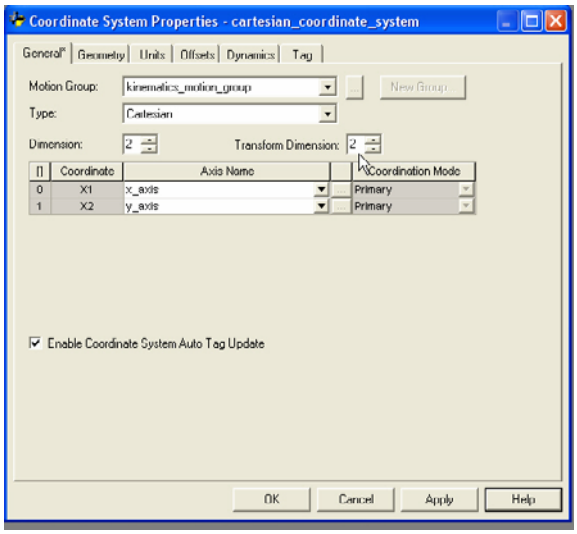
You can also use the Coordinate System Properties dialog boxes to edit an existing Coordinate System tag. These have a series of tabs that access a specific dialog box for configuring the different facets of the Coordinate System. Make the appropriate entries for each of the fields. An asterisk appears on the tab to indicate changes have been made but not implemented. Click Apply to save your selections.

TIP When you configure your coordinate system, some fields can be unavailable (dimmed) because of choices you made in the New Tag dialog box.

In the Controller Organizer, right-click the coordinate system to edit and choose Coordinate System Properties from the pull-down menu.



The Coordinate System Properties General dialog box appears.



The name of the Coordinate System tag that is being edited appears in the title bar to the right of Coordinate System Properties.

General Tab

Use this tab to do the following for a coordinate system:

- Assign the coordinate system, or terminate the assignment of a coordinate system, to a Motion Group.
- Choose the type of coordinate system you are configuring.
- Change the number of dimensions, that is, the number of axes.
- Specify the number of axes to transform.
- Assign axes to the coordinate system tag.
- Enable/Disable automatic updating of the tag.

Logix Designer application supports only one Motion Group tag per controller.

Table 3 - General Tab Field Descriptions

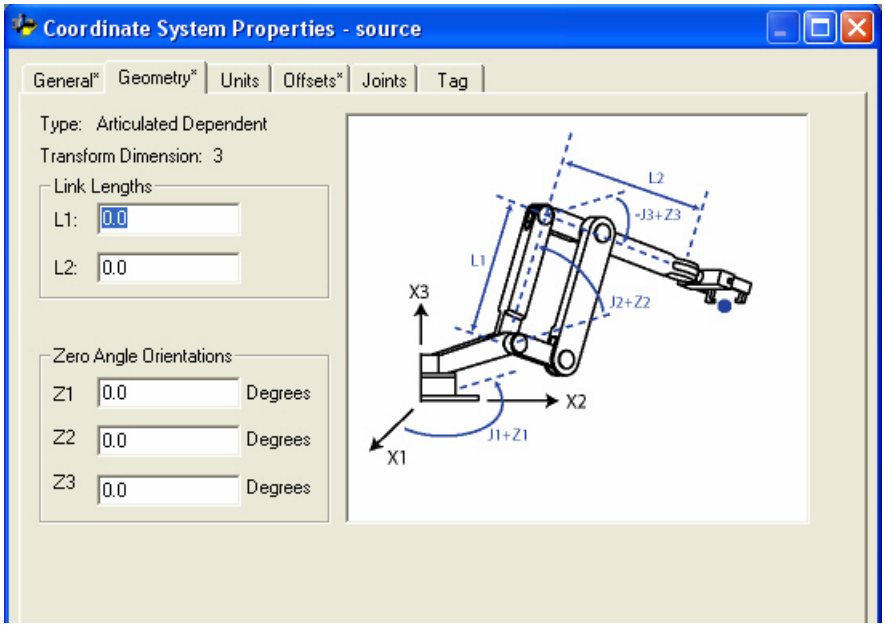
Item	Description
Motion Group	Motion Group is where you can select and display the Motion Group to which the Coordinate System is associated. A Coordinate System assigned to a Motion Group appears in the Motion Groups branch of the Controller Organizer, under the selected Motion Group sub-branch. Selecting <none> terminates the Motion Group association, and moves the coordinate system to the Ungrouped Axes sub-branch of the Motions Groups branch.
Ellipsis (…)	Ellipsis opens the Motion Group Properties dialog box for the Assigned Motion Group where you can edit the Motion Group properties. If no Motion Group is assigned to this coordinate system, this is unavailable.
New Group	New Group opens the New Tag dialog box where you can create a new Motion Group tag. This is enabled only if no Motion Group tag has been created.
Type	Type selects and displays the type of coordinate system (robot type) in the Motion Group. Available choices are Cartesian, Articulated Dependent, Articulated Independent, SCARA Independent, Delta, and SCARA Delta. The type of coordinate system you choose in this field changes the configuration tabs that are available.
Dimension	Enter the coordinate system dimensions, that is, the number of axes, that this coordinated system is to support. The options are 1, 2, or 3 in keeping with its support of a maximum of three axes. Changes in the Dimension spin also reflect in the Axis Grid by either expanding or contracting the number of fields available. Data is set back to the defaults for any axis that is removed from the Axis Grid due to reducing the Dimension field.
Transform Dimension	Enter the number of axes in the coordinate system that you want to transform. The options are 1, 2, or 3 in keeping with its support of a maximum of 3 axes. The number of axes that you transform must be equal to or less than the specified coordinate system dimensions. The transform function always begins at the first axis. For example, if you have specified that the coordinate system has 3 axes, but indicate only that 2 axes be transformed, then axes 1 and 2 are transformed. In other words, you cannot specify that only axes number 2 and number 3 be transformed.
Axis Grid	The Axis Grid is where you associate axes to the Coordinate System. There are five columns in the Axis Grid that provide information about the axes in relation to the Coordinate System.
[] (Brackets)	The Brackets column displays the indices in tag arrays used with the current coordinate system. The tag arrays used in multi-axis coordinated motion instructions map to axes by using these indices.
Coordinate	The text in this column X1, X2, or X3 (depending on the entry to the Dimension field) is used as a cross reference to the axes in the grid. For a Cartesian system, the mapping is simple.

Table 3 - General Tab Field Descriptions

Item	Description
Axis Name	The Axis Name column is a list of combo boxes (the number is determined by the Dimension field) used to assign axes to the coordinate system. The pull-down lists display all of the Base Tag axes defined in the project. (Alias Tag axes do not display in the pull-down list.) They can be axes associated with the motion group, axes associated with other coordinated systems, or axes from the Ungrouped Axes folder. Choose an axis from the pull-down list. The default is <none>. It is possible to assign fewer axes to the coordinate system than the maximum for the Dimension field; however, you receive a warning when you verify the coordinate system and, if left in that state, the instruction generates a run-time error. You can assign an axis only once in a coordinate system. Ungrouped axes also generate a runtime error.
Ellipsis (...)	The Ellipsis in this column takes you to the Axis Properties pages for the axis listed in the row.
Coordination Mode	The Coordination Mode column indicates the axes that are used in the velocity vector calculations. If the type of coordinate system is specified as Cartesian, then Primary axes are used in these calculations. For non-Cartesian coordinate systems, the coordination mode for the axes defaults to Ancillary.
Enable Coordinate System Auto Tag Update	The Enable Coordinate System Auto Tag Update checkbox lets you determine whether the Actual Position values of the current coordinated system are automatically updated during operation. Use the checkbox to enable this feature. The Coordinate System Auto Tag Update feature can ease your programming burden if you need to add GSV statements to the program to get the desired result. However, by enabling this feature, the Coarse Update rate is increased. Whether to use the Coordinate System Auto Tag Update feature depends upon the trade-offs between ease in programming and increase in execution time. You can lower the execution time if you enable this feature in initial system programming to work out the kinks and then disable it and enter the GSV statements in your program. Enabling this feature can result in some performance penalty.

Geometry Tab

The Geometry tab of the Coordinate System Properties is where you can specify the link lengths and zero angle orientation values for articulated robotic arms.



The graphic displayed on this tab shows a typical representation of the type of coordinate system you selected on the General tab. Your robot typically looks similar to the one shown in the graphic, but can be somewhat different depending on your application.

Link Lengths Box

The Link Length box displays fields to let you specify a value for the length of each link in an articulated robotic arm (coordinate system). The measurement units for the articulated coordinate system are defined by the measurement units configured for the affiliated Cartesian coordinate system. The two coordinate systems are linked or affiliated with each other by an MCT instruction.

When specifying the link length values, be sure that the values are calculated by using the same measurement units as the linked Cartesian coordinate system. For example, if the manufacturer specifies the robot link lengths by using millimeter units and you want to configure the robot by using inches, then you must convert the millimeter link measurements to inches and enter the values in the appropriate link length fields.

IMPORTANT	Be sure that the link lengths specified for an articulated coordinate system are in the same measurement units as the affiliated Cartesian coordinate system. Your system does not work properly if you are using different measurement units.

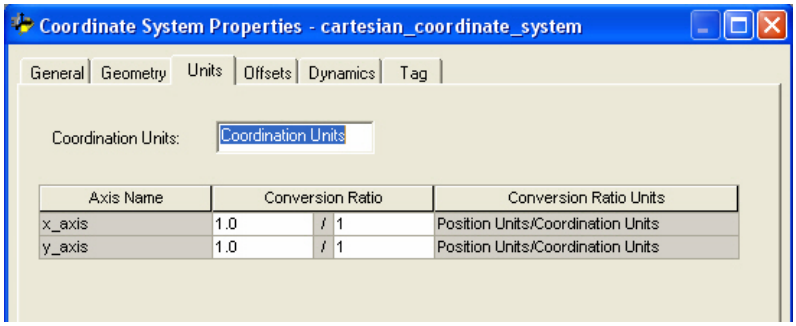
The number of fields available for configuration in the link lengths box is determined by values entered on the General tab for the type of coordinate system, total coordinate system dimensions, and transform dimensions. The link identifiers are L1 and L2 in the corresponding graphic. These fields are not configurable for a Cartesian coordinate system.

Zero Angle Orientations Box

The zero-angle orientation is the rotational offset of the individual joint axes. If applicable, enter the offset value in degrees for each joint axis. The number of available fields is determined by the coordinate dimension value entered on the General tab. The angle identifiers are Z1, Z2, and Z3 in the corresponding graphic.

Units Tab

The Units tab of the Coordinate System Properties is where you determine the units that define the coordinate system. This dialog box is where you define the Coordination Units and the Conversion Ratios.



Coordination Units

The Coordination Units field lets you define the units to be used for measuring and calculating motion related values such as position and velocity. The coordination units do not need to be the same for each coordinate system. Enter units that are relevant to your application and maximize ease of use. When you change the Coordination Units, the second portion of the Coordination Ratio Units automatically changes to reflect the new units. Coordination Units is the default.

Axis Grid

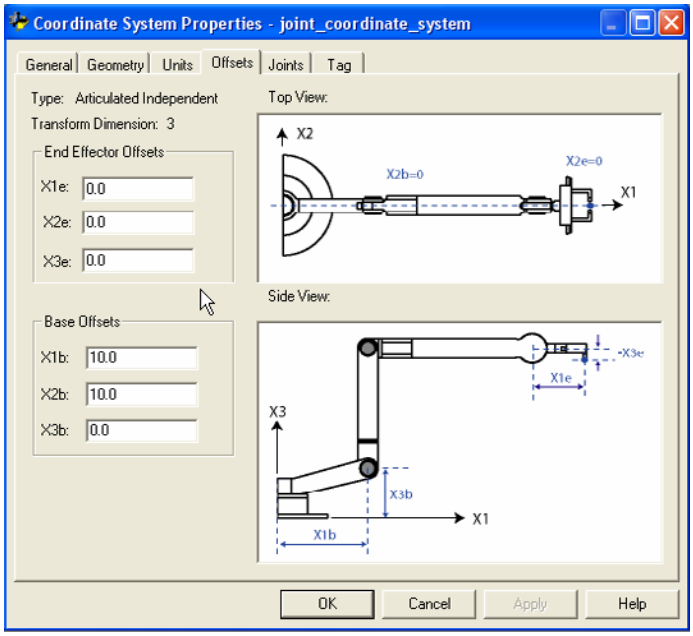
The Axis Grid of the Units dialog box displays the axis names associated with the coordinate system, the conversion ratio, and the units used to measure the conversion ratio.

Table 4 - Units Tab Description

Item	Description
Axis Name	The Axis Name column contains the names of the axes assigned to the coordinate system in the General dialog box. These names appear in the order that they were configured into the current coordinate system. You cannot edit this column from this dialog box.
Conversion Ratio	The Conversion Ratio column defines the relationship of axis position units to coordination units for each axis. For example, if the position units for an axis is in millimeters and the axis is associated with a coordinate system whose units are in inches, then the conversion ratio for this axis/coordinate system association is 25.4/1 and can be specified in the appropriate row of the Axis Grid. The numerator can be entered as a float or an integer. The denominator must be entered only as an integer.
Conversion Ratio Units	The Conversion Ratio Units column displays the axis position units to coordination units used. The Axis Position units are defined in the Axis Properties – Units dialog box and the coordination units are defined in Coordinated System Properties – Units dialog box. These values are dynamically updated when changes are made to either axis position units or coordination units.

Offsets Tab

The Offsets tab of the Coordinate System Properties dialog box is where you define the end effector and base offset values for the robotic arm. This tab shows the top and/or sides view of a typical robotic arm based on the type of coordinate system and coordinate Transform dimension values specified on the General tab. The number of available offset fields in each box is determined by the number of axes associated with the coordinate system.



When specifying the end effector and base offset values, be sure that the values are calculated by using the same measurement units as the linked Cartesian coordinate system.

For example, if the manufacturer specifies the robot offset by using millimeter units and you want to configure the robot by using inches, then you must convert the millimeter link measurements to inches and enter the values in the appropriate offset fields.

End Effector Offsets Box

The end effector offset value specifies the dimensions of the end effector. The correct end effector offsets are typically available from the manufacturer. The end effector indicators are X1e, X2e, and X3e in the corresponding graphic.

Base Offsets Box

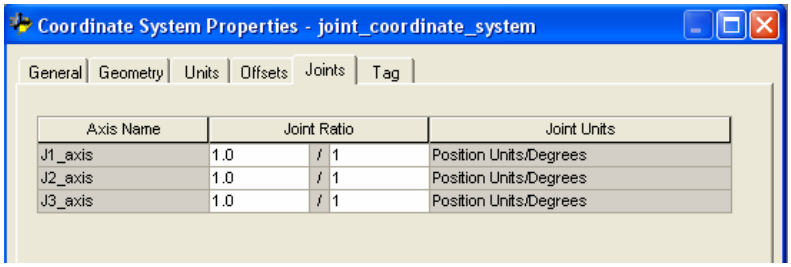
The Logix Designer Kinematics internal equations define the robot origin relative to the first joint of the robotic arm. Sometimes the robot manufacturer specifies the origin at a different location. The difference between these two locations is the base offsets value. The correct base offset values are typically available from the robot manufacturer. The base offset indicators are X1b, X2b, and X3b in the corresponding graphic.

Joints Tab

The Joints tab is accessible only if you are configuring or editing an articulated coordinate system. This dialog box is where you define the Joint Conversion Ratios. Joint axis units are always specified in degrees.

Table 5 - Joints Tab Field Descriptions

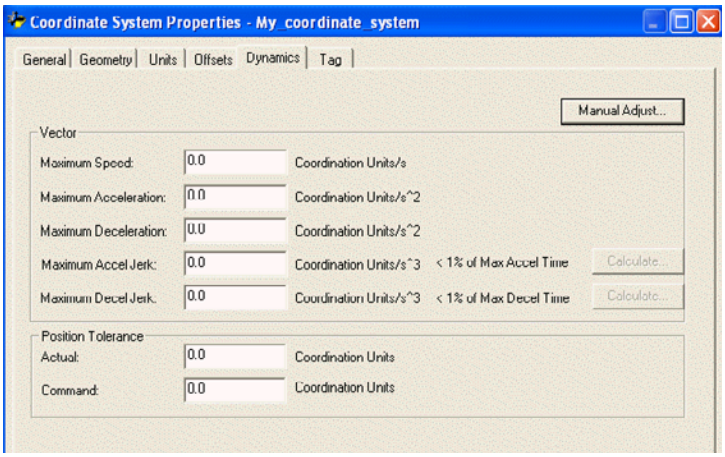
Item	Description
Axis Name	The Axis Name column displays the names of the axes associated to the coordinate system. The names appear in the order that they were configured into the coordinate system. This is a read-only field.
Joint Ratio	The Joint Ratio column (shown in white) is divided into two columns that define the relationship between the axis position units to the joint axis units. The left-half of the Joint Ratio column is a configurable field that lets you specify a value for the axis position units (numerator). The right-half of the Joint Ratio column is a configurable field that lets you specify a value for the joint axis units (denominator). Keep in mind that Joint axis units are always specified as degrees.
Joint Units	The Joint Units column is a read-only field that displays the configured axis position units to the joint units. The Axis Position units are defined in the Axis Properties – Units dialog box. Joint units are always defined as degrees.



If you are configuring a Cartesian coordinate system, go to the Dynamics tab to access the Coordinate System Properties Dynamics dialog box.

Dynamics Tab

The Dynamics dialog box is accessible only if you are configuring a Cartesian coordinate system. The Dynamics tab is for entering the Vector values used for Maximum Speed, Maximum Acceleration, Maximum Deceleration, Maximum Acceleration Jerk and Maximum Deceleration Jerk. It is also used for entering the Actual and Command Position Tolerance values.



Vector Box

In the Vector box, values are entered for Maximum Speed, Maximum Acceleration, Maximum Deceleration, Maximum Acceleration Jerk, and Maximum Deceleration Jerk. The values are used by the Coordinated Motion instructions in calculations when their operands are expressed as percent of Maximum. The Coordination Units to the right of the edit boxes automatically change when the coordination units are redefined in the Units dialog box.

Table 6 - Dynamics Tab Field Descriptions

Item	Description
Maximum Speed	Enter the value for Maximum Speed to be used by the Coordinated Motion instructions in calculating vector speed when speed is expressed as a percent of maximum.
Maximum Acceleration	Enter the value for Maximum Acceleration to be used by the Coordinated Motion instructions to determine the acceleration rate to apply to the coordinate system vector when acceleration is expressed as a percent of maximum.
Maximum Deceleration	Enter the value for Maximum Deceleration to be used by the Coordinated Motion instructions to determine the deceleration rate to apply to the coordinate system vector when deceleration is expressed as a percent of maximum. The Maximum Deceleration value must be a nonzero value to achieve any motion by using the coordinate system.

Table 6 - Dynamics Tab Field Descriptions

Item	Description
Maximum Acceleration Jerk	The jerk parameters only apply to S-Curve profile moves by using these instructions: • MCS • MCCD • MCCM • MCLM The Maximum Acceleration Jerk rate of the coordinate system, in Coordination Units/second^3, defaults to 100% of the maximum acceleration time. The speed and acceleration rate for this calculation are defined above. $\frac{\text{MaxAccel}^2}{\text{Speed}}$ = Maximum Acceleration Jerk The Maximum Accel Jerk value entered is used when the motion instruction is set with Jerk Units=% of Maximum. When a Multi-axis Motion Instruction has Jerk Units=units per sec^3, then the maximum acceleration jerk value is derived from the motion instruction faceplate. The jerk units for the motion instruction also allow for Jerk Units=% of Time, with 100% of Time. This means that the entire S-Curve move has Jerk limiting. This is the default mode. An S-Curve move with 0% of Time results in a trapezoidal profile and have 0% Jerk limiting. If set manually, enter the value in units=Coordination Units/second^3 units. You can also use the Calculate button to view this value in terms of units=% of Time.
Maximum Deceleration Jerk	The jerk parameters only apply to S-Curve profile moves by using these instructions: • MCS • MCCD • MCCM • MCLM The Maximum Deceleration Jerk rate of the coordinate system, in Coordination Units/second^3, defaults to 100% of the maximum deceleration time. The speed and deceleration rate for the calculation are defined above. $\frac{\text{MaxDecel}^2}{\text{Speed}}$ = Maximum Deceleration Jerk The Maximum Decel Jerk value entered is used when the motion instruction is set with Jerk Units=% of Maximum. When a Multi-axis motion instruction has Jerk Units=units per sec^3, then the Max Deceleration Jerk value is derived from the Motion Instruction faceplate. The jerk units for the motion instruction also allow for Jerk Units=% of Time, with 100% of Time meaning the entire S-Curve move has Jerk limiting, thus, the default mode. An S-Curve move with 0% of Time results in a trapezoidal profile and has 0% Jerk limiting. If set manually, enter the value in units=Coordination Units/second^3 units. You can also use the optional Calculate button to view the value in terms of units=% of Time.

Position Tolerance Box

In the Position Tolerance Box, values are entered for Actual and Command Position Tolerance values. See the Logix5000 Motion Controllers Instructions Reference Manual, publication MOTION-RM002 and Appendix B of this manual for more information regarding the use of Actual and Command Position Tolerance.

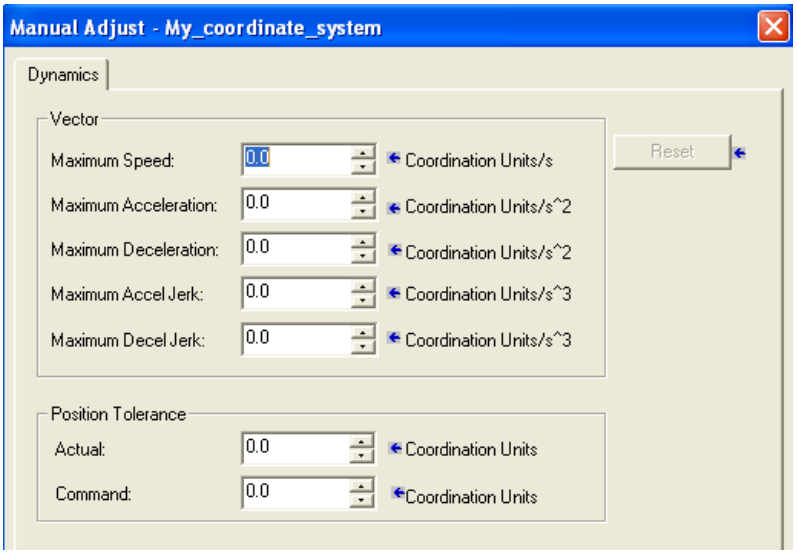
Item	Description
Actual	Enter the value in coordination units for Actual Position to be used by Coordinated Motion instructions when they have a Termination Type of Actual Tolerance.
Command	Enter the value in coordination units for Command Position to be used by Coordinated Motion instructions when they have a Termination Type of Command Tolerance.

Manual Adjust Button

The Manual Adjust button on the Coordinate System Dynamics tab accesses the Manual Adjust Properties dialog box. The Manual Adjust button is enabled only when there are no pending edits on the properties dialog box.

Dynamics Tab Manual Adjust

At this dialog box you can make changes to the Vector and Position Tolerance values.



These changes can be made either online or offline. The blue arrows to the right of the fields indicate that they are immediate commit fields. This means that the values in those fields are immediately updated to the controller if online or to the project file if offline.

Reset

Reset reloads the values that were present at the time this dialog box was entered. The blue arrow to the right of Reset means that the values are immediately reset when you click Reset.

Motion Planner Tab

The Motion Planner dialog box is accessible only if you are configuring a Cartesian coordinate system. The Motion Planner tab is used to enable or disable Master Delay Compensation, enable or disable Master Position Filter, and to enter the bandwidth for Master Position Filter.

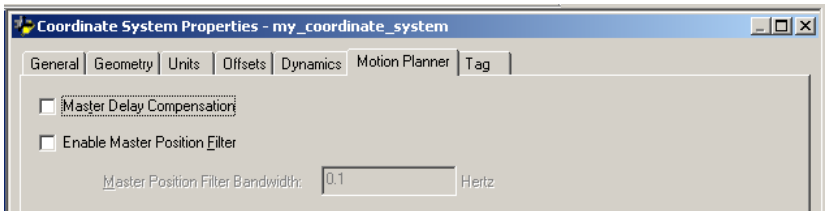
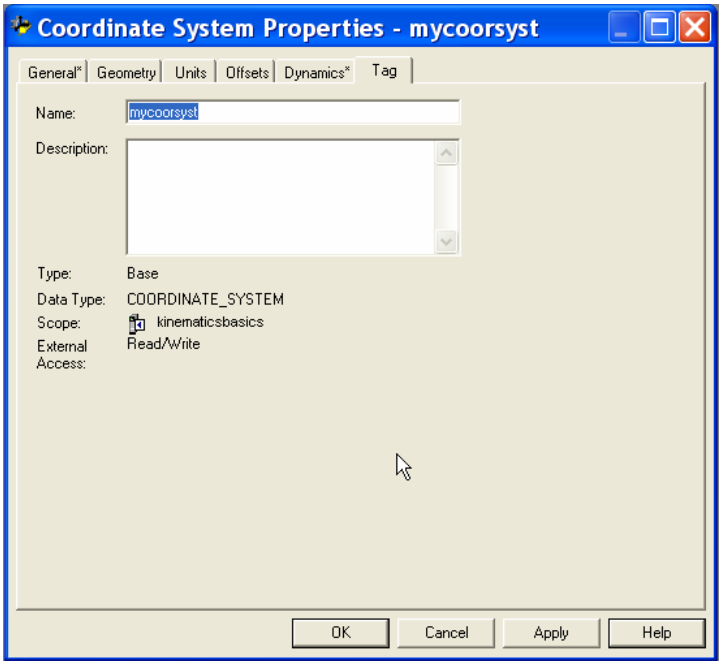


Table 7 - Motion Planner Tab Field Descriptions

Item	Description
Master Delay Compensation	Check or clear this box to enable or disable Master Delay Compensation, respectively. This value is used to balance the delay time between reading the Master Axis comm position and applying the associated slave command position to the slave's servo loop. This feature ensures that the slave coordinate command position accurately tracks the actual position of the Master Axis (that is, zero tracking error when gearing or camming to the actual position of a Master Axis for Cartesian coordinate motion in Master Driven mode). Clear this box to disable Master Delay Compensation. The default setting is Enabled. If the axis is configured for Feedback only, disable Master Delay Compensation. In some applications, there is no requirement for zero tracking error between the Master and the Slave axis. In these cases, it can be beneficial to disable the Master Delay Compensation feature to eliminate the disturbances introduced to the Slave Axis. Note that Master Delay Compensation, even if the box is checked, is not applied in cases where a Slave Axis is gearing or camming to the Master Axis' command position because there is no need to compensate for master position delay.
Enable Master Position Filter	Check or clear this box to enable or disable Master Position Filter, respectively. The default is cleared (disabled). Master Position Filter, when enabled, effectively filters the specified master axis position input to the slave axis's gearing or position camming operation. The filter smooths out the actual position signal from the Master Axis, and thus smooths out the corresponding motion of the Slave Axis. When this box is checked, the Master Position Filter Bandwidth box is enabled.
Master Position Filter Bandwidth	The Master Position Filter Bandwidth field is enabled when the Enable Master Position Filter check box is checked. This field controls the bandwidth for master position filtering. Enter a value in Hz in this field to set the bandwidth for the Master Position Filter. Note that a value of zero for Master Position Filter Bandwidth effectively disables the master position filtering.

Tag Tab

The Tag tab is for reviewing your Tag information and renaming the tag or editing the description.



Use this tab to modify the name and description of the coordinate system. When you are online, all of the parameters on this tab transition to a read-only state, and cannot be modified. If you go online before you save your changes, all pending changes revert to their previously-saved state.

Table 8 - Tag Tab Field Descriptions

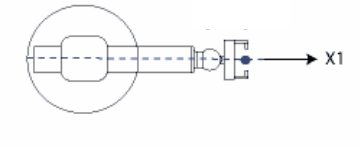
Item	Description
Name	Name displays the name of the current tag. You can rename the tag at this time. The name can be up to 40 characters and can include letters, numbers, and underscores (_). When you rename a tag, the new name replaces the old one in the Controller Organizer after you click OK or Apply.
Description	Description displays the description of the current tag, if any is available. You can edit this description. The edited description replaces the existing description when you click OK or Apply.
Tag Type	Tag Type indicates the type of the current Coordinate System tag. This type can be either a base or an alias. The field is not editable and is for informational purposes only.
Data Type	Data Type displays the data type of the current Coordinate System tag, which is always COORDINATE_SYSTEM. This field cannot be edited and is for informational purposes only.
Scope	Scope displays the scope of the current Coordinate System tag. The scope for a Coordinate System tag can be only controller scope. This field is not editable and is for informational purposes only.
External Access	External Access displays the parameter chosen in the New Tag dialog box for whether the tag has Read/Write, Read Only, or no (None) access from external applications such as HMIs.

Cartesian Coordinate System

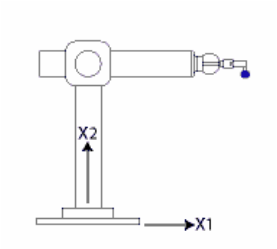
Topic	Page
Program an MCLM Instruction	34
Blended Moves and Termination Types	34
Bit State Diagrams for Blended Moves	37
Choose a Termination Type	41
MCLM and MCCM Examples	48

Use the multi-axis coordinated motion instructions to perform linear and circular moves in single and multidimensional spaces. A Cartesian coordinate system in Logix Designer application can include one, two or three axis.

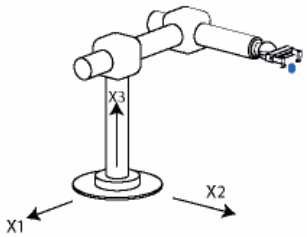
Figure 3 - Coordinate Systems with Orthogonal Axes



Cartesian Coordinate System



Two-dimensional Cartesian Coordinate System



Three-dimensional Cartesian Coordinate System

Use the MCLM instruction to start a single or multi-dimensional linear coordinated move. See Motion Coordinated Linear Move (MCLM) on page 109.

Use the MCCM instruction to initiate a two or three-dimensional circular coordinated move for the specified axes. See Motion Coordinated Circular Move (MCCM) on page 136.

Program an MCLM Instruction

The following are the steps to program and test an MCLM instruction.

1. Set up motion axes in Logix Designer application.

   The maximum number of axes that can be associated with one Coordinate System is limited to three axes.

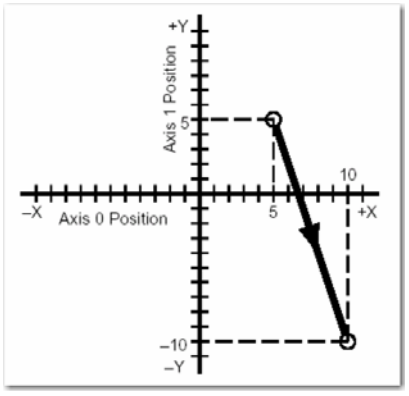
2. Create a Coordinate System Tag

   The number of Coordinate System tags that can be created is 32. This number is based on the fact that a maximum of 32 axes can be assigned to a motion group and in the current implementation. Because only one motion group can be created, the number of axes that can be created is 32.

3. Program an MCLM.

   The Motion Coordinated Linear Move (MCLM) instruction performs a linear move by using up to three axes in a Cartesian coordinate system. As with all moves, you must specify, for example, absolute or incremental, or speed. Speed is based on the vector move distance as shown below.

$$V_{distance} = \sqrt{5^2 + 15^2}$$



   Position is defined by a single dimension array. Array length is determined by the coordinate system selected. For a (2) Axis Cartesian System, each endpoint requires (2) words; for a (3) Axis Cartesian System, each axis requires (3) words. We'll create a position array very shortly for clarification. An array can consist of multiple endpoint coordinates that can be used by multiple coordinated move instructions.

Blended Moves and Termination Types

To blend two MCLM or MCCM instructions, start the first one and queue the second one. The tag for the coordinate system gives you two bits for queueing instructions.

- MovePendingStatus
- MovePendingQueueFullStatus

The MCLM and MCCM instructions reference a coordinate system called Coordinate_System_1 (cs1). For example, the following ladder diagram uses coordinate system cs1 to blend Move1 into Move2.

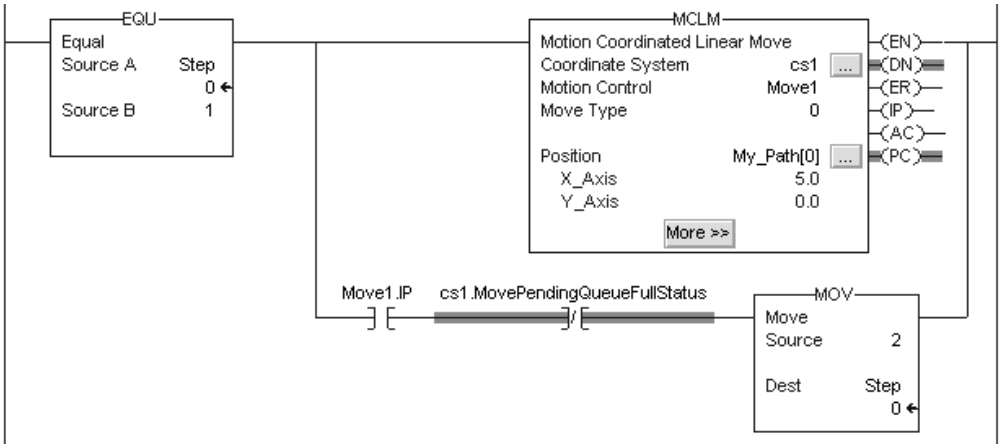
Example Ladder Diagram for Blended Instructions

If Step = 1, then:

Move1 starts and moves the axes to a position of 5, 0.

and once Move1 is in process and there is room to queue another move, then:

Step = 2.



If Step = 2, then:

Move1 is already happening.

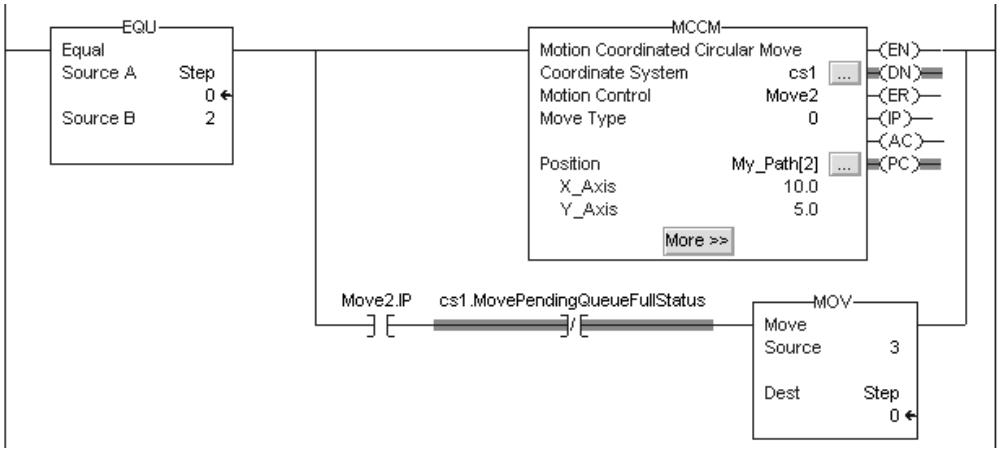
Move2 goes into the queue and waits for Move1 to complete.

When Move1 is complete:

Move2 moves the axes to a position of 10, 5.

And once Move2 is in process and there is room in the queue:

Step = 3.



When an instruction completes, it is removed from the queue and there is space for another instruction to enter the queue. Both bits always have the same value because you can queue only one pending instruction at a time. If the application requires several instructions to be executed in sequence, then the bits are set by using these parameters.

Table 9 - Bit Parameters

When	Then
One instruction is active and a second instruction is pending in the queue	• MovePendingStatus bit = 1 • MovePendingQueueFullStatus bit = 1 • You cannot queue another instruction
An active instruction completes and leaves the queue	• MovePendingStatus bit = 0 • MovePendingQueueFullStatus bit = 0 • You can queue another instruction

The termination type operand for the MCLM or MCCM instruction specifies how the currently executing move gets terminated. These illustrations show the states of instruction bits and coordinate system bits that get affected at various transition points (TP).

The termination types are:

- 0 - Actual tolerance
- 1 - No Settle
- 2 - Command Tolerance
- 3 - No Decel
- 4 - Follow Contour Velocity Constrained
- 5 - Follow Contour Velocity Unconstrained
- 6 - Command Tolerance Programmed

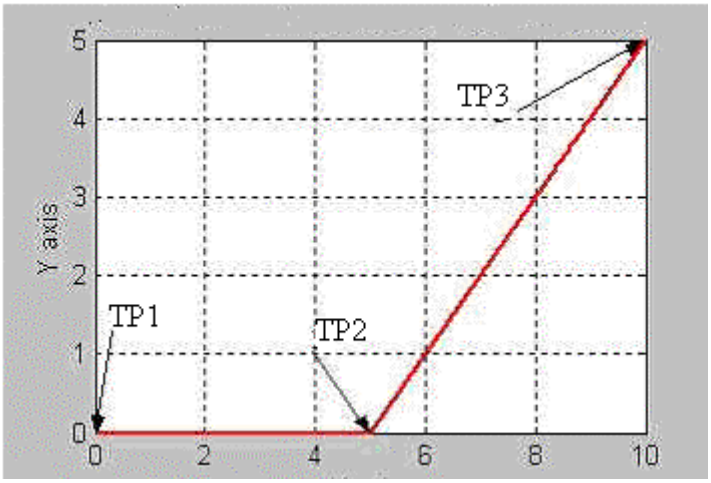
For further information on how to select a termination type, refer to Choose a Termination Type on page 41.

Bit State Diagrams for Blended Moves

The following diagrams show bit states at the transition points for various types of blended moves.

Bit States at Transition Points of Blended Move by Using Actual Tolerance or No Settle

linear → linear move



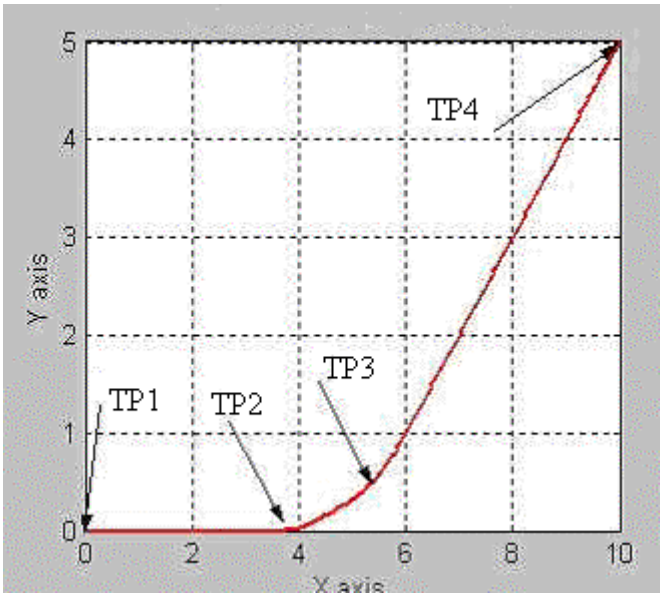
This table shows the bit status at the various transition points shown in the preceding graph with termination type of either Actual Tolerance or No Settle.

Table 10 - Bit Status at Transition Points with Actual Tolerance or No Settle Termination Type

Bit	TP1	TP2	TP3
Move1.DN	T	T	T
Move1.IP	T	F	F
Move1.AC	T	F	F
Move1.PC	F	T	T
Move2.DN	T	T	T
Move2.IP	T	T	F
Move2.AC	F	T	F
Move2.PC	F	F	T
cs1.MoveTransitionStatus	F	F	F
cs1.MovePendingStatus	T	F	F
cs1.MovePendingQueueFullStatus	T	F	F

Bit States at Transition Points of Blended Move by Using No Decel

linear → linear move



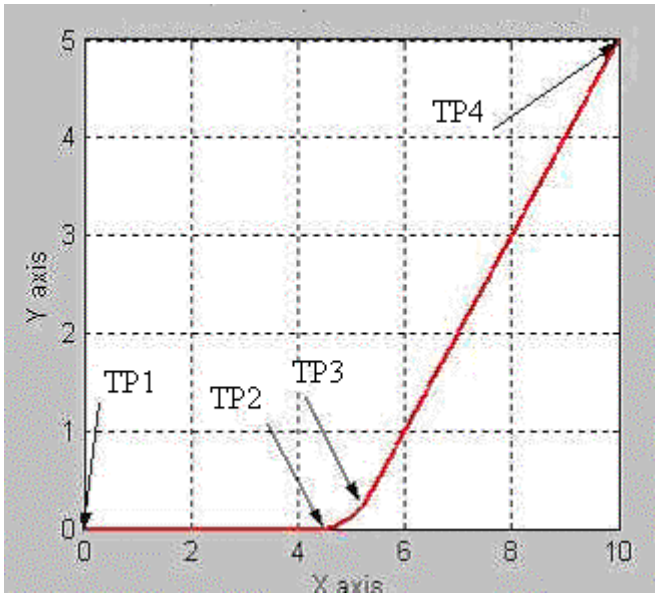
This table shows the bit status at the various transition points shown in the preceding graph with termination type of No Decel. For No Decel termination type distance-to-go for transition point TP2 is equal to deceleration distance for the Move1 instruction. If Move 1 and Move 2 are collinear, then Move1.PC is true at TP3 (the programmed end-point of first move).

Table 11 - Bit Status with No Decel Termination Type

Bit	TP1	TP2	TP3	TP4
Move1.DN	T	T	T	T
Move1.IP	T	F	F	F
Move1.AC	T	F	F	F
Move1.PC	F	T	T	T
Move2.DN	T	T	T	T
Move2.IP	T	T	T	F
Move2.AC	F	T	T	F
Move2.PC	F	F	F	T
cs1.MoveTransitionStatus	F	T	F	F
cs1.MovePendingStatus	T	F	F	F
cs1.MovePendingQueueFullStatus	T	F	F	F

Bit States at Transition Points of Blended Move by Using Command Tolerance

linear → linear move



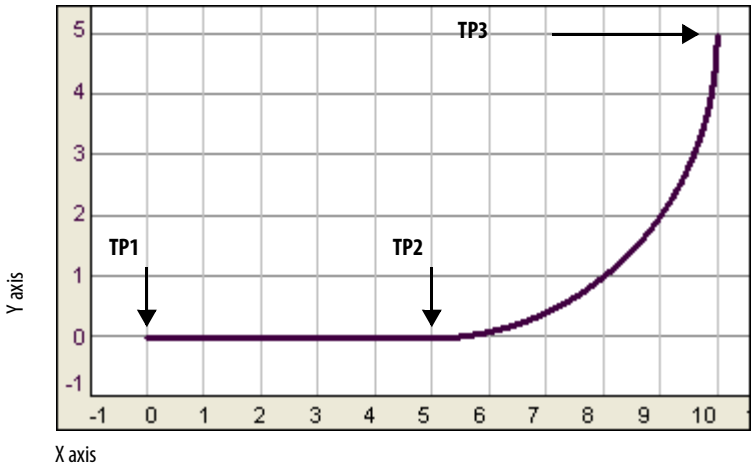
This table shows the bit status at the various transition points shown in the preceding graph with termination type of Command Tolerance. For Command Tolerance termination type distance-to-go for transition point TP2 is equal to Command Tolerance for the coordinate system cs1.

Table 12 - Bit Status with Command Tolerance Termination Type

Bit	TP1	TP2	TP3	TP4
Move1.DN	T	T	T	T
Move1.IP	T	F	F	F
Move1.AC	T	F	F	F
Move1.PC	F	T	T	T
Move2.DN	T	T	T	T
Move2.IP	T	T	T	F
Move2.AC	F	T	T	F
Move2.PC	F	F	F	T
cs1.MoveTransitionStatus	F	T	F	F
cs1.MovePendingStatus	T	F	F	F
cs1.MovePendingQueueFullStatus	T	F	F	F

Bit States at Transition Points of Blended Move by Using Follow Contour Velocity Constrained or Unconstrained

linear → circular move



This table shows the bits status at the transition points.

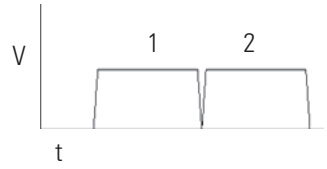
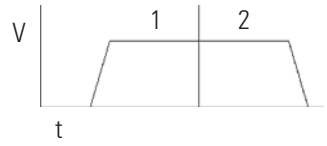
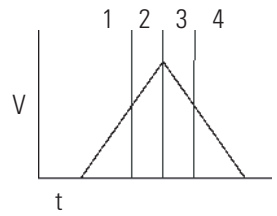
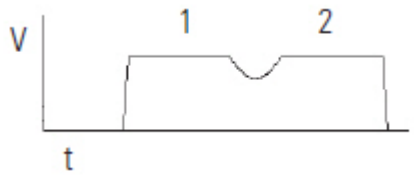
Table 13 - Bit Status with Contour Velocity Constrained or Unconstrained Termination Type

Bit	TP1	TP2	TP3
Move1.DN	T	T	T
Move1.IP	T	F	F
Move1.AC	T	F	F
Move1.PC	F	T	T
Move2.DN	T	T	T
Move2.IP	T	T	F
Move2.AC	F	T	F
Move2.PC	F	F	T
cs1.MoveTransitionStatus	F	F	F
cs1.MovePendingStatus	T	F	F
cs1.MovePendingQueueFullStatus	T	F	F

Choose a Termination Type

The termination type determines when the instruction is complete. It also determines how the instruction blends its path into the queued MCLM or MCCM instruction, if there is one.

1. Choose a termination type.

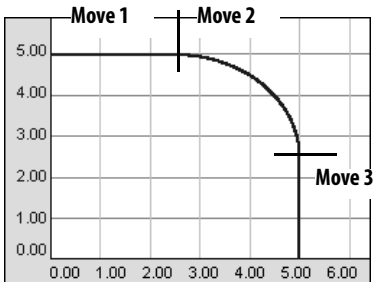
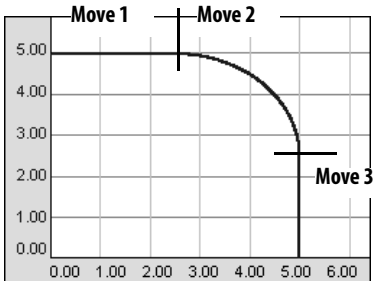
If You Want the Axes to (vector speeds)	And You Want the Instruction to Complete When the	Then Use this Termination Type
Stop between moves 	Both of these happen: • Command position equals target position. • The vector distance between the target and actual positions is less than or equal to the Actual Position Tolerance of the coordinate system.	0 - Actual Tolerance
	Command position equals the target position.	1 - No Settle
Keep the speed constant **except** between moves	Command position gets within the Command Position Tolerance of the coordinate system.	2 - Command Tolerance
	Axes get to the point at which they must decelerate at the deceleration rate.	3 - No Decel
Transition into or out of a circle without stopping 		4 - Follow Contour Velocity Constrained
Accelerate or decelerate across multiple moves 		5 - Follow Contour Velocity Unconstrained
Use a specified Command Tolerance 	The command position gets within the Command Position Tolerance of the coordinate system.	6 - Command Tolerance Programmed

2. Make sure this is the right choice for you.

Termination Type	Example Path	Description
0 - Actual Tolerance		The instruction stays active until both of these happen: • Command position equals target position. • The vector distance between the target and actual positions is less than or equal to the Actual Position Tolerance of the coordinate system. At that point, the instruction is complete and a queued MCLM or MCCM instruction can start. **Important**: Make sure that you set the Actual Tolerance to a value that your axes can reach. Otherwise the instruction stays in process.
1 - No Settle		The instruction stays active until the command position equals the target position. At that point, the instruction is complete and a queued MCLM or MCCM instruction can start.
2, 6- Command Tolerance		The instruction stays active until the command position gets within the Command Tolerance of the coordinate system. At that point, the instruction is complete and a queued MCLM or MCCM instruction can start. If you don't have a queued MCLM or MCCM instruction, the axes stop at the target position.

Logix Designer Application Compares	To the	And Uses the	For the
100% of the configured length of the first instruction by using a Command Tolerance termination type	Configured Command Tolerance for the coordinate system	Shorter of the two lengths	Command Tolerance length used for the **first** instruction
100% of the configured length of the last move instruction by using a Command Tolerance termination type	Configured Command Tolerance for the coordinate system	Shorter of the two lengths	Command Tolerance length used for the **next to last** instruction
50% of each of the lengths of all other move instructions	Configured Command Tolerance for the coordinate system	Shorter of the two lengths	Command Tolerance length used for **each individual** instruction

3 - No Decel		The instruction stays active until the axes get to the deceleration point. At that point, the instruction is complete and a queued MCLM or MCCM instruction can start. • The deceleration point depends on whether you use a trapezoidal or S-Curve profile. • If you don't have a queued MCLM or MCCM instruction, the axes stop at the target position.

Termination Type	Example Path	Description
4 - Follow Contour Velocity Constrained		The instruction stays active until the axes get to the target position. At that point, the instruction is complete and a queued MCLM or MCCM instruction can start. • This termination type works best with tangential transitions. For example, use it to go from a line to a circle, a circle to a line, or a circle to a circle. • The axes follow the path. • The length of the move determines the maximum speed of the axes. If the moves are long enough, the axes do not decelerate between moves. If the moves are too short, the axes decelerate between moves.
5 - Follow Contour Velocity Unconstrained		This termination type is similar to the contour velocity constrained. It has these differences: • Use this termination type to get a triangular velocity profile across several moves. This reduces jerk. • To avoid position overshoot at the end of the last move, you must calculate the deceleration speed at each transition point during the deceleration-half of the profile. • You must also calculate the starting speed for each move in the deceleration half of the profile.

Important Considerations

If you stop a move by using an MCS or by changing the speed to zero with an MCCD during a blend and then resume the move by reprogramming the move or by using an another MCCD, it deviates from the path that displayed if the move had not been stopped and resumed. The same phenomenon can occur if the move is within the decel point of the start of the blend. In either case, the deviation is most likely a slight deviation.

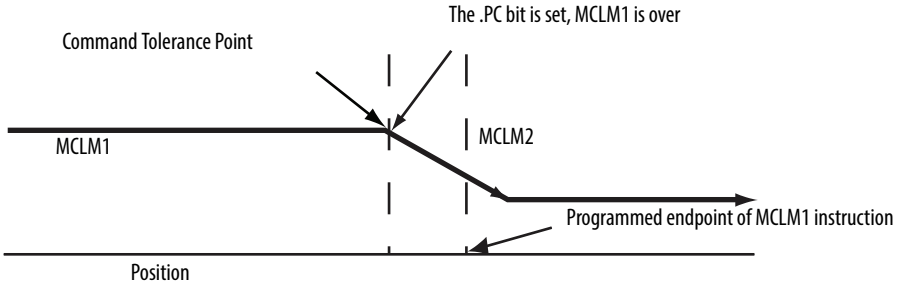
Velocity Profiles for Collinear Moves

Collinear moves are those that lie on the same line in space. Their direction can be the same or opposite. The velocity profiles for collinear moves can be complex. This section provides you with examples and illustrations to help you understand the velocity profiles for collinear moves programmed with MCLM instructions.

Velocity Profiles for Collinear Moves with Termination Type 2 or 6

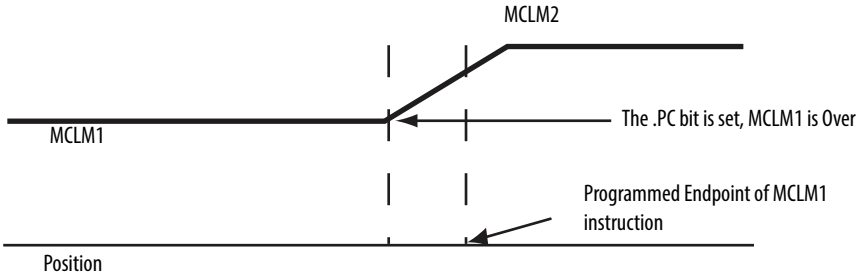
This illustration shows the velocity profile of two collinear moves by using a Command Tolerance (2) termination type. The second MCLM instruction has a **lower** velocity than the first MCLM instruction. When the first MCLM instruction reaches its Command Tolerance point, the move is over and the .PC bit is set.

Figure 4 - Velocity Profile of Two Collinear Moves When the Second Move has a Lower Velocity than the First Move and Termination Type 2 or 6 is Used



This illustration show the velocity profile of two collinear moves by using a Command Tolerance (2) termination type. The second MCLM instruction has a **higher** velocity than the first MCLM instruction. When the first MCLM instruction reaches its Command Tolerance point, the move is over and the .PC bit is set.

Figure 5 - Velocity Profile of Two Collinear Moves When the Second Move has a Higher Velocity than the First Move and Termination Type 2 or 6 is Used



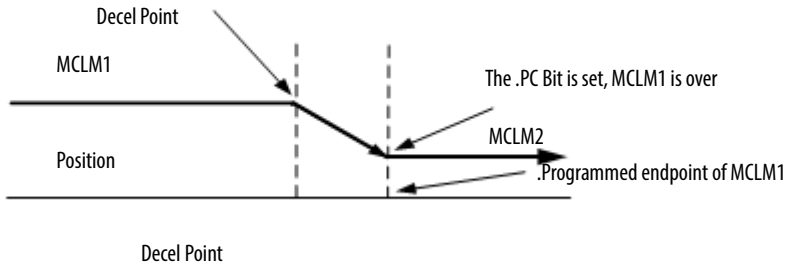
Velocity Profiles for Collinear Moves with Termination Types 3, 4, or 5

This illustration shows a velocity profile of two collinear moves. The second MCLM instruction has a **lower** velocity than the first MCLM instruction and one of these termination types are used:

- No Decel (3)
- Follow Contour Velocity Constrained (4)
- Follow Contour Velocity Unconstrained (5)

When the first MCLM instruction reaches the deceleration point, it decelerates to the programmed velocity of the second move. The first move is over and the .PC bit is set.

Figure 6 - Velocity Profile of Two Collinear Moves When the Second Move has a Lower Velocity than the First Move and Termination Type 3, 4, or 5 is Used

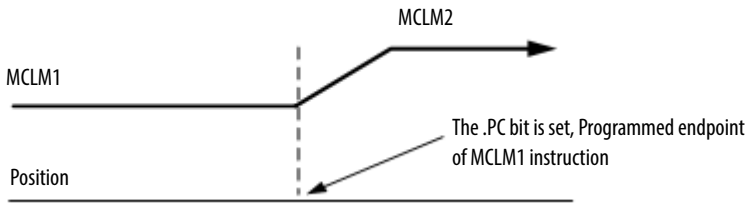


This illustration shows a velocity profile of two collinear moves. The second MCLM instruction has a **higher** velocity than the first MCLM instruction and one of these termination types are used:

- No Decel (3)
- Follow Contour Velocity Constrained (4)
- Follow Contour Velocity Unconstrained (5)

The .PC bit is set when the first move reaches its programmed endpoint.

Figure 7 - Velocity Profile of Two Collinear Moves When the Second Move has a Higher Velocity than the First Move and Termination Type 3, 4, or 5 is Used



Symmetric Profiles

Profile paths are symmetric for all motion profiles.

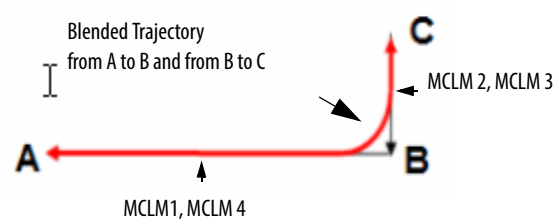
Programming the velocity, acceleration, and deceleration values symmetrically in the forward and reverse directions generates the same path from point A to point C in the forward direction, as from point C to point A in the reverse direction.

While this concept is most easily shown in a two-instruction sequence, it applies to instruction sequences of any length provided that they are programmed symmetrically.

Figure 8 - Example of a Symmetric Profile

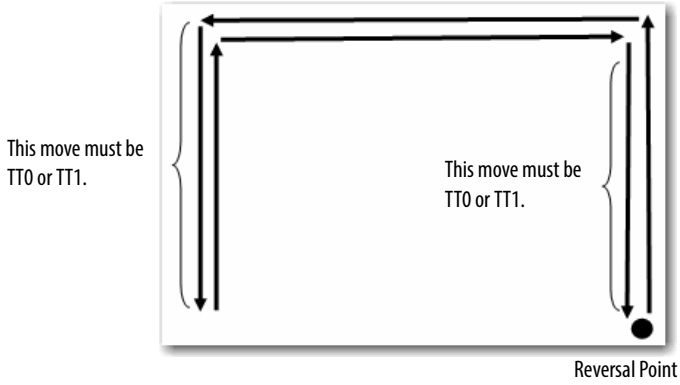
- MCLM 1 (point A to point B) is followed by MCLM 2 (point B to point C).
- MCLM 3 (point C to point B) is followed by MCLM 4 (point B to point A).
- The acceleration of MCLM 1 must be equal to the deceleration of MCLM 4.
- The deceleration of MCLM 1 must be equal to the acceleration a MCLM 4.
- The acceleration of MCLM 2 must be equal to the deceleration of MCLM 3.
- The deceleration of MCLM 2 must be equal to the acceleration of MCLM 3.

MCLM 1 (Pos = [2,0], Accel = 1, Decel = 2)
MCLM 2 (Pos = [2,1], Accel = 3, Decel = 4)
MCLM 3 (Pos = [2,0], Accel = 4, Decel = 3)
MCLM 4 (Pos = [0,0], Accel = 2, Decel = 1)



IMPORTANT	We recommend that you terminate any sequence of moves by either Termination Type 0 or 1, that is, TT0 or TT1.

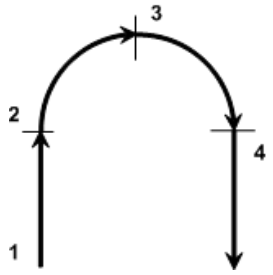
To guarantee that your trajectory is symmetric, you must terminate any sequence of moves by either Termination Types 0 or 1. Use a Termination Type of 0 or 1 at the Reversal Point of a profile that moves back on itself.



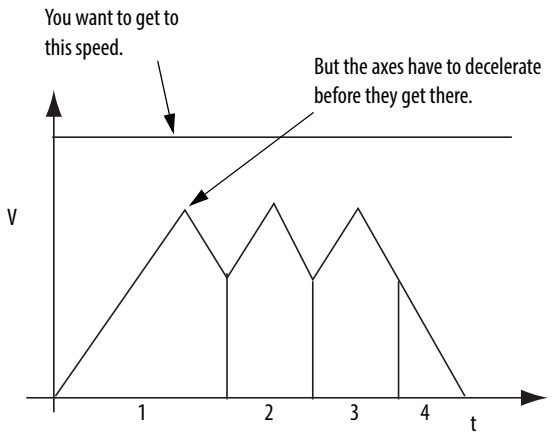
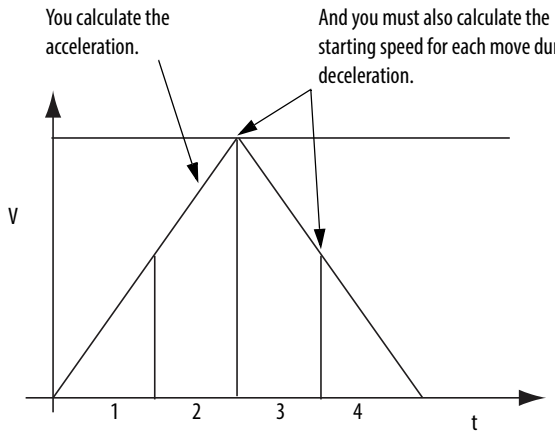
Using a TT2, TT3, TT4, TT5 or TT6 as the last move in a profile (or the reversal point) is safe. However, the resulting trajectory from A to B cannot always be the same as that from B to A. Explicit termination of the sequence of moves helps the controller to optimize the velocity profile, reduce the CPU load, and guarantee a symmetric profile.

Triangular Velocity Profile

If you want to program a pick and place action in four moves, minimize the Jerk rate, and use a triangular velocity profile.

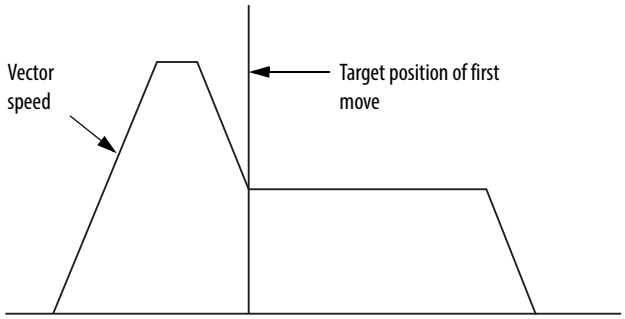
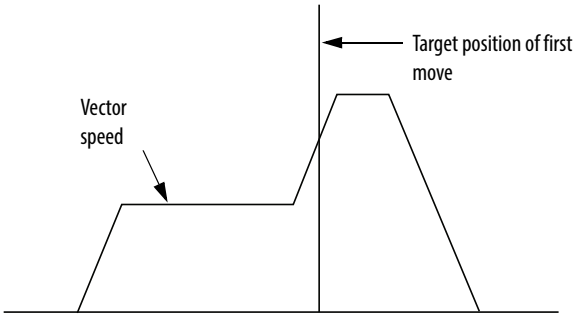
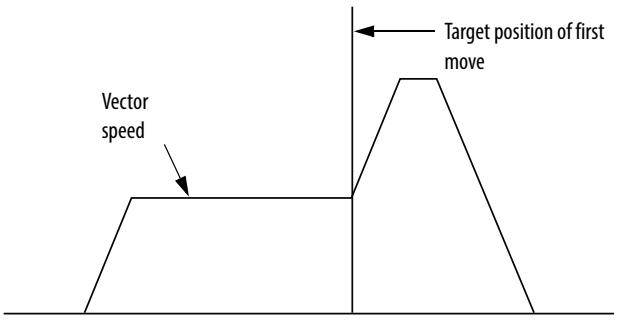


Then, use termination type 5. The other termination types can prevent you from getting to the speed you want.

Termination Types 2, 3, 4, or 6	Termination Type 5
	
The length of each move determines its maximum speed. As a result, the axes do not reach a speed that causes them to overshoot the target position during deceleration.	The axes accelerate to the speed that you want. You must calculate the starting speed for each move in the deceleration-half of the profile.

Blending Moves at Different Speeds

You can blend MCLM and MCCM instructions where the vector speed of the second instruction is different from the vector speed of the first instruction.

If the Next Move is	And the Termination Type of the First Move is	Then
Slower	2 - Command Tolerance 3 - No Decel 4 - Contour Velocity Constrained 5 - Contour Velocity Unconstrained 6 - Command Tolerance Programmed	
Faster	2 - Command Tolerance 3 - No Decel 6 - Command Tolerance Programmed	
	4 - Contour Velocity Constrained 5 - Contour Velocity Unconstrained	

MCLM and MCCM Examples

Refer to Motion Coordinated Instructions (MCLM, MCCM, MCCD, MCS, MCSD, MCT, MCTP, MCSR, MDCC) on page 109.

Kinematics Coordinate Systems

Topic	Page
Useful Terms	50
Gather Information about Your Robot	50
Summary of Kinematic Steps	51
Determine the Coordinate System Type	52

This chapter provides you with the information you need when using the Kinematics functionality within Logix Designer application. This chapter also provides you with guidelines for robot-specific applications.

Kinematics coordinate systems use two instructions, the Motion Calculate Transform Position (MCTP) and the Motion Coordinate Shutdown Reset (MCSR).

Motion Calculate Transform Position (MCTP)

Use the MCTP instruction to calculate the position of a point in one coordinate system to the equivalent point in a second coordinate system.

Motion Coordinated Shutdown Reset (MCSR)

Use the Motion Coordinated Shutdown Reset (MCSR) instruction to reset all axes in a coordinate system. The MCSR instruction resets the axes from a shutdown state to an axis ready state. This instruction also clears any axis faults.

ATTENTION: Use each tag for the motion control attribute of instructions only once. Re-use of the motion control tag in other instructions can cause unintended operation. This can result in damage to equipment or personal injury.

Useful Terms

Understanding the terms used in this chapter enables you to properly configure your robot.

Term	Definition
Forward Kinematics	The solution of source positions given the target positions. In practice, requires computing the Cartesian positions given the Joint positions.
Forward Transform	The solution of source positions given target positions.
Inverse Kinematics	The solution of joint positions given Cartesian positions. Typically, converts Cartesian positions to joint positions.
Inverse Transform	The solution of target positions given source positions.
Joint axis	A rotary robotic coordinate axis typically having overtravel rather than rollover limits.
Kinematics	The family of mathematical equations that convert positions back and forth between two linked geometries.
Orientation	Robotic term for directional attitude or rotation about a point in Cartesian (3D) space. Orientation is expressed as three ordered rotations around the X, Y, and Z Cartesian axes.
Reference frame	An imaginary Cartesian coordinate system used to define a Cartesian origin and reference orientation.
Source system	One of two coordinate systems used in a Kinematics transform and having special properties. When connected to a target system by means of a Kinematics transform, motion commanded at the source system's inputs produces motion at both the source and target system's outputs (if the physical axes are connected).
Target system	One of two coordinate systems used in a Kinematics transform and having special properties. When connected to a source system by means of a Kinematics transform, motion commanded at the target system's inputs produces motion in both the source and target system's outputs (if the physical axes are connected).
Tool Center Point	All Kinematics programmed position (motion) is based on the Tool Center Point (TCP). To determine the TCP, you must enter information on these Logix Designer application tabs: • Geometry - Enter values for Link Length (linear displacement), Zero Angle Orientation (angular rotation), and Base Offsets. These values, in combination with the selected Geometry type, defines the resulting Geometry's end-of-arm position. • Offsets - Enter value for End-effector offset; these are included when establishing the final TCP position.
Transform	General term for conversion equations that map values in one coordinate space to values in another coordinate space.
Translation	Robotic term for a linear movement or offset in Cartesian (three-dimensional) space. Translation describes the distance between two Cartesian points.
Zero Angle Offset	Offset on a rotary axis in the Joint Coordinate system between where the Kinematics equations were derived and where you want your zero position to be.

Gather Information about Your Robot

Before you begin the configuration steps for the Kinematics transformation function, you need to gather specific information about your robot and application parameters. Specifications for your robot can be found in the documentation provided by the manufacturer; other required information is application dependent. You need to know this information before you begin configuring motion control.

- Robot geometry type
- Zero angle orientation
- Work envelope
- Link lengths
- Base offsets
- End-effector offsets
- Arm solution

Summary of Kinematic Steps

After you create a Joint (target) coordinate system tag for your Motion control project, there are general steps to follow for Kinematics.

1. Determine and then configure the type of coordinate system you need for your robot.

   For help in determining your coordinate system type, see page 52.

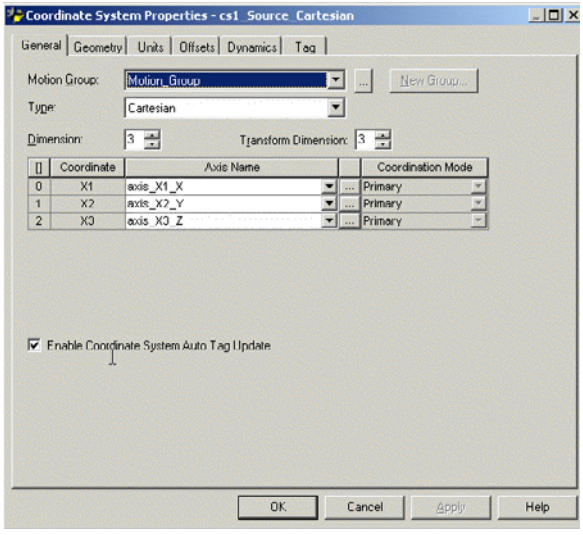
2. Establish the Joint-to-Cartesian reference frame relationship.

   For more information regarding the joint-to-Cartesian reference frame, see the section about the type of robot you are using.

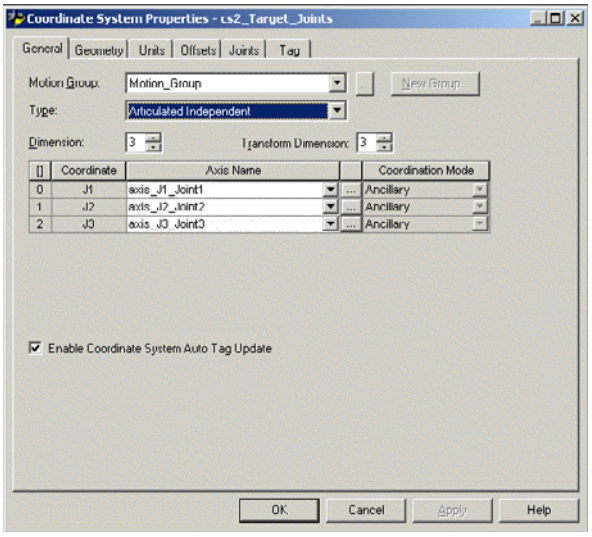
WARNING: The correct relationship between the Joint reference frame and the Cartesian reference frame must be established. Failure to do this can allow your robot to move to unexpected positions causing machine damage and/or injury or death to personnel.

3. Calibrate your robot (if applicable).
4. Identify your robot work envelope.
5. Determine and then configure the following parameters:
   - Link lengths
   - Base offsets
   - End-effector offsets
6. Create the source and target coordinate systems.

Typical Cartesian Coordinate System Configuration for Articulated Independent robot.



Typical Joint Coordinate System Configuration for an Articulated Independent robot.



7. Save the project.

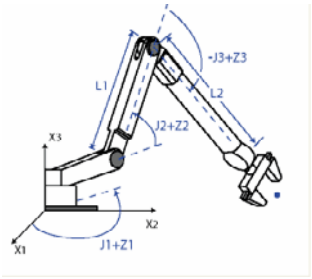
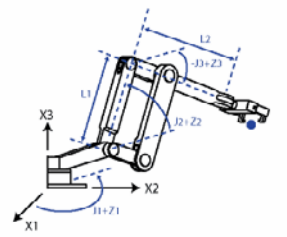
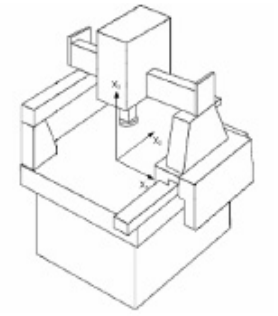
8. Download the Kinematic project to the controller and then use the MCT instruction to link the Joint coordinate system to the Cartesian coordinate system.

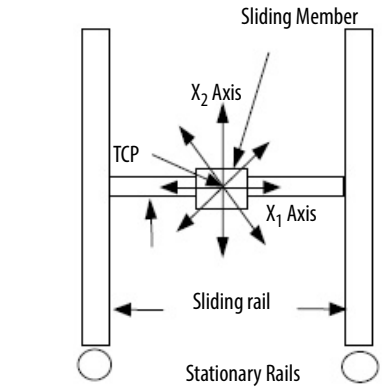
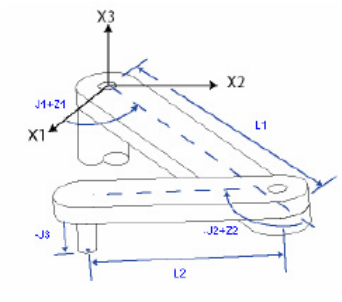
   The Joint-to-Cartesian reference frame relationship is automatically established by the controller after the Joint coordinate system parameters (link lengths, base offsets, and end-effector offsets) are configured and the MCT instruction is enabled. For additional information about the MCT or MCTP instructions, see the Logix5000 Controllers Motion Instructions, publication MOTION-RM002.

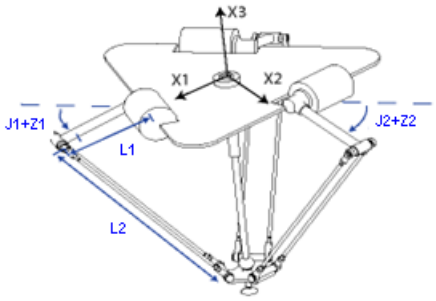
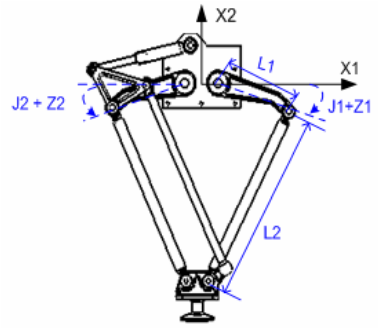
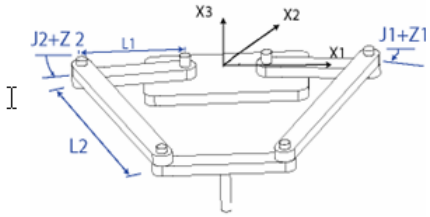
For detailed steps about Creating and Configuring a Coordinate System, see on Create and Configure a Coordinate System page 15.

Determine the Coordinate System Type

Use this table to determine what type of Kinematics coordinate system you need.

If your robot looks similar to	Your Coordinate System type is
	Articulated Independent For configuration information, see page 55.
	Articulated Dependent For configuration information, see page 93.
	Cartesian This illustration shows a typical Gantry machine. For configuration information, see page 103.

If your robot looks similar to	Your Coordinate System type is
	Cartesian This illustration shows a typical H-bot. For configuration information see page 105.
	SCARA Independent For configuration information, see page 86.

If your robot looks similar to	Your Coordinate System type is
	Three-dimensional Delta For configuration information, see page 65.
	Two-dimensional Delta For configuration information, see page 73.
	SCARA Delta For configuration information, see page 78.

Articulated Independent Robot

Topic	Page
Reference Frame	55
Methods to Establish a Reference Frame	57
Work Envelope	59
Configuration Parameters	61
Delta Robot Geometries	64
Configure a Delta Three-dimensional Robot	65
Configure a Delta Two-dimensional Robot	73
Configure a SCARA Delta Robot	78
Arm Solutions	83
Configure a SCARA Independent Robot	86
Error Conditions	90

Use these guidelines when configuring an Articulated Independent robot.

WARNING: Before turning ON the Transform and/or establishing the reference frame, be sure to do the following for the joints of the target coordinate system.

1. Set and enable the soft travel limits.
2. Enable the hard travel limits.

Failure to do this can allow the robot to move outside of the work envelope causing machine damage and/or serious injury or death to personnel.

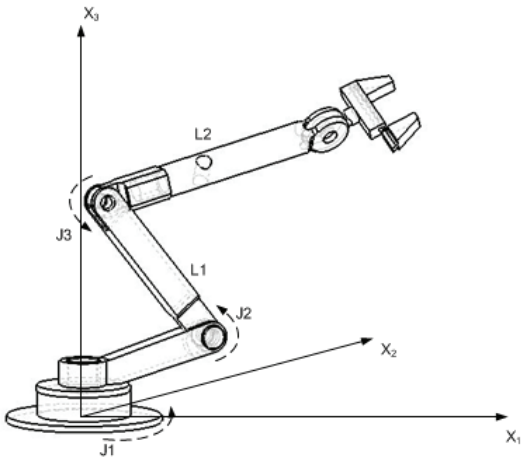
Reference Frame

The reference frame is the Cartesian coordinate frame that defines the origin and the three primary axes (X1, X2, and X3). These axes are used to measure the real Cartesian positions.

WARNING: Failure to properly establish the correct reference frame for your robot can cause the robotic arm to move to unexpected positions causing machine damage and/or injury or death to personnel.

The reference frame for an Articulated Independent robot is at the base of the robot, as shown in Figure 1.

Figure 9 - Articulated Independent 1



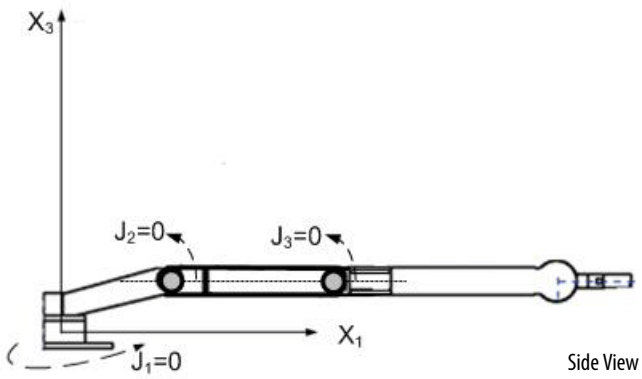
Before you begin establishing the Joint-to-Cartesian reference frame relationship, it is important to know some information about the Kinematic mathematical equations used in the controllers. The equations were written as if the Articulated Independent robot joints were positioned as shown in this figure.

- +J1 is measured counterclockwise around the +X3 axis starting at an angle of J1=0 when L1 and L2 are both in the X1-X2 plane.
- +J2 is measured counterclockwise starting with J2=0 when L1 is parallel to X1-X2 plane.
- +J3 is measured counterclockwise with J3=0 when L2 is aligned with link L1.

When your robot is physically in this position, the Logix Designer application Actual Position tags for the axes must be:

- J1 = 0.
- J2 = 0.
- J3 = 0.

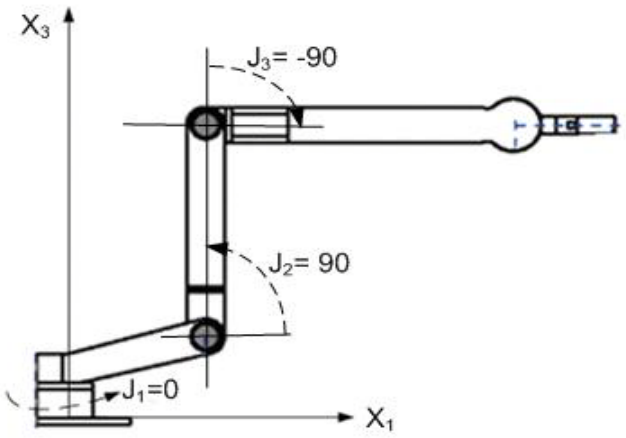
Figure 10 - Articulated Independent 2



When your robot is physically in this position, the Logix Designer application Actual Position tags for the axes must be:

- J1 = 0.
- J2 = 90.
- J3 = -90.

Figure 11 - Articulated Independent 3



Side View

If your robot's physical position **and** joint angle values cannot match those shown in either figures above, then use one of the Alternate Methods for Establishing the Joint-to-Cartesian reference frame relationship.

Methods to Establish a Reference Frame

The following methods let you establish a reference frame for an Articulated Independent robot.

For each	Use one of these methods to establish the reference frame
Incremental axis	Each time the robot's power is cycled.
Absolute axis	Only when you establish absolute home.

- Method 1 - establishes a Zero Angle Orientation and lets the configured travel limits and home position on the joint axes remain operational. Use this method if you are operating the axes between the travel limits determined prior to programming a Motion Redefine Position (MRP) instruction and want these travel limits to stay operational.
- Method 2 - uses a MRP instruction to redefine the axes position to align with the Joint reference frame. This method can require the soft travel limits to be adjusted to the new reference frame.

Method 1 - Establishing a Reference Frame

Each axis for the robot has the mechanical hard stop in each of the positive and negative directions. Manually move or press each axes of the robot against its associated mechanical hard stop and redefine it to the hard limit actual position provided by the robot manufacturer. J1 is the axis at the base of the robot that rotates around X3.

When the robot is moved so that Link1 is parallel to the X3 axis and Link2 is parallel to X1 axis as shown in Articulated Independent 3 on page 57, the Logix Designer application Actual Position tag values are equal to:

- J1 = 0.
- J2 = 90°.
- J3 = -90°.

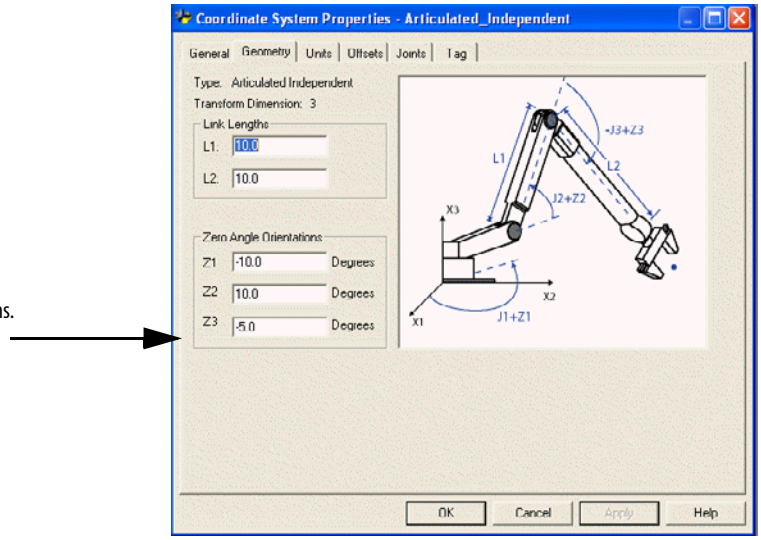
If the Logix Designer application Positions tags do not correspond to these values, configure the Zero Angle Orientation for the joint or joints that do not correspond.

If the Logix Designer application read-out values are	Set the Zero Angle Orientations on the Coordinate System Properties dialog to
J1 = 10 J2 = 80 J3 = -85	Z1 = -10 Z2 = 10 Z3 = -5

The Joint-to-Cartesian reference frame relationship is automatically established by the ControlLogix controller after the Joint coordinate system parameters (link lengths, base offsets, and end-effector offsets) are configured and the MCT instruction is enabled.

Figure 12 - Setting the Zero Angle Orientations

Set the Zero Angle Orientations.



Method 2 - Establishing a Reference Frame

Position the robot so that:

- Link1 is parallel to the X3 axis.
- Link2 is parallel to X1 axis.

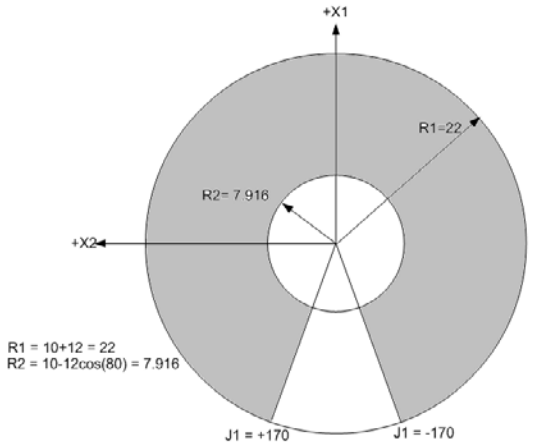
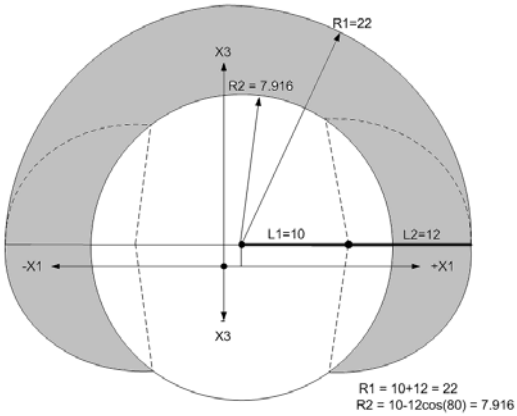
Program a MRP instruction for all three axes with the following values:

- J1 = 0
- J2 = 90°
- J3 = -90°

The Joint-to-Cartesian reference frame relationship is automatically established by the ControlLogix controller after the Joint coordinate system parameters (link lengths, base offsets, and end-effector offsets) are configured and the MCT instruction is enabled.

Work Envelope

The work envelope is the three-dimensional region of space that defines the reaching boundaries for the robot arm. The work envelope for an articulated robot is ideally a complete sphere having an inner radius equal to L1- L2 and outer radius equal to L1+L2. Due to the range of motion limitations on individual joints, the work envelope is not always a complete sphere.

If the range-of-motion values for the articulated robot are	Typically, the work envelope is
J1 = ± 170 J2 = 0 to 180 J3 = ± 100 L1= 10 L2 = 12	 Top view - Depicts the envelope of the tool center point sweep in J1 and J3 while J2 remains at a fixed position of 0°.  Side view - Depicts the envelope of the tool center point sweep in J2 and J3 while J1 remains at a fixed position of 0°.

Configuration Parameters

Logix Designer application can be configured for control of robots with varying reach and payload capacities. As a result, it is very important to know the configuration parameter values for your robot including:

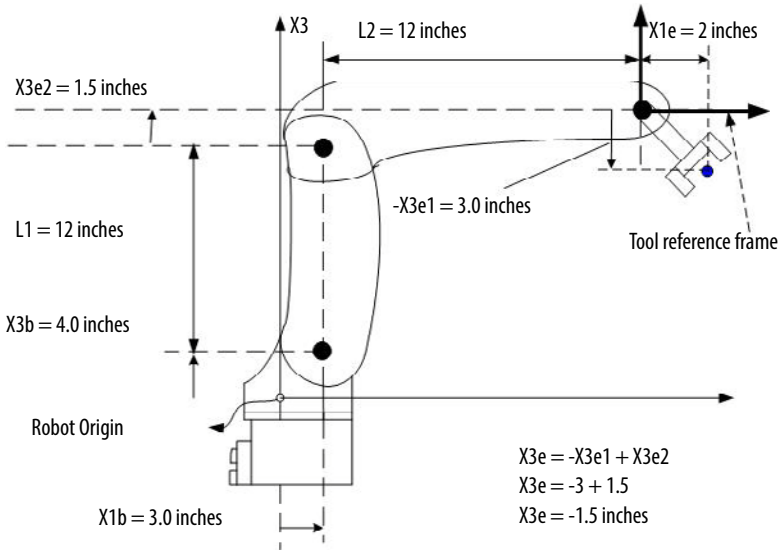
- Link lengths.
- Base offsets.
- End-effector offsets.

The configuration parameter information is available from the robot manufacturer.

IMPORTANT	Verify that the values for the link lengths, base offsets and end-effector offsets entered into the Configuration Parameters dialog use the same measurement units.

This example illustrates the typical configuration parameters for an Articulated Independent robot.

Figure 13 - Typical Configuration Parameters for an Articulated Independent Robot



If the robot is two-dimensional, then X3b and X3e are X2b and X2e respectively.

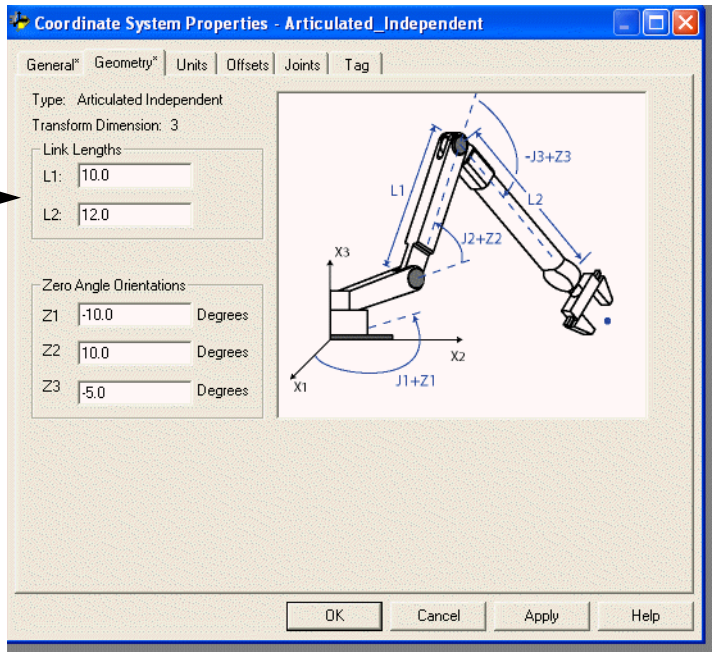
Link Lengths

Link lengths are the rigid mechanical bodies attached at joints.

For an articulated independent robot with	The length of	Is equal to the value of the distance between
2 dimensions	L1 L2	J1 and J2 J2 and the end-effector
3 dimensions	L1 L2	J2 and J3 J3 and the end-effector

Figure 14 - Example of Link Lengths for an Articulated Independent Robot

Enter the Link Length values.



For the robot shown in Typical Configuration Parameters for an Articulated Independent Robot, the Link Length values are:

- L1 = 10.0
- L2 = 12.0

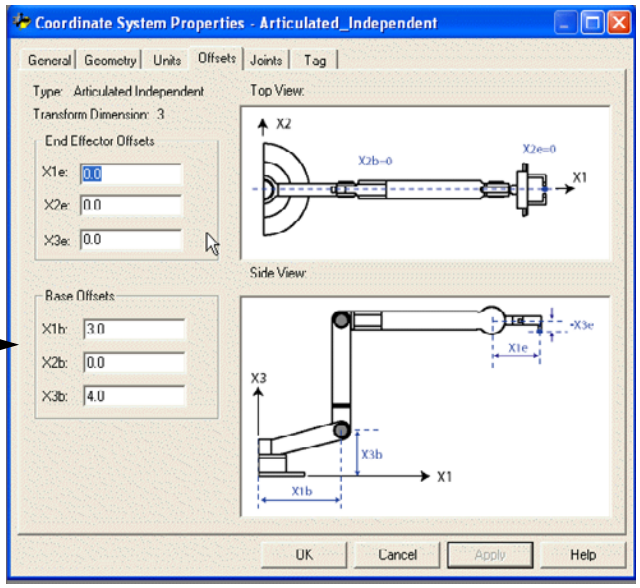
Base Offsets

The base offset is a set of coordinate values that redefines the origin of the robot. The correct base offset values are typically available from the robot manufacturer. Enter the values for the base offsets in the X1b and X3b fields of the Coordinate System Properties dialog.

Figure 15 - Example of Base Offsets for an Articulated Independent Robot

Enter the Base Offset values.
For the robot shown in our example, the Base Offset values are:

- X1b = 3.0
- X3b = 4.0



End-effector Offsets

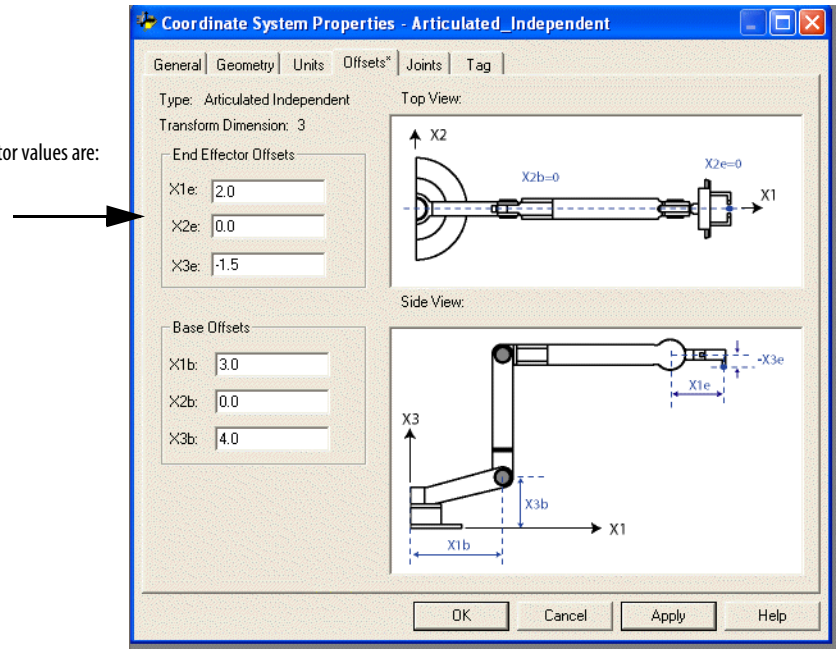
The robot can have an end-effector attached to the end of robot link L2. If there is an attached end-effector, then you must configure the end-effector offset value on the Coordinate System Properties dialog. The end-effector offsets are defined with respect to the tool reference frame at the tool tip.

Some robots also have an offset defined for the J3 joint as shown in the robot example Typical Configuration Parameters for an Articulated Independent Robot on page 61. You can account for this value when computing the X3e end-effector offset value. In Typical Configuration Parameters for an Articulated Independent Robot, the value for X3e offset is entered as the sum of X3e1+X3e2 (-3+1.5 = -1.5). The configured value for X3e is -1.5.

Figure 16 - Example of End-effectors for an Articulated Independent Robot

Enter the end-effector offset values.
For the robot shown in our example, the end-effector values are:

- X1e = 2.0
- X3e = -1.5



Delta Robot Geometries

Logix Designer application supports three types of geometries that are often called parallel manipulators.

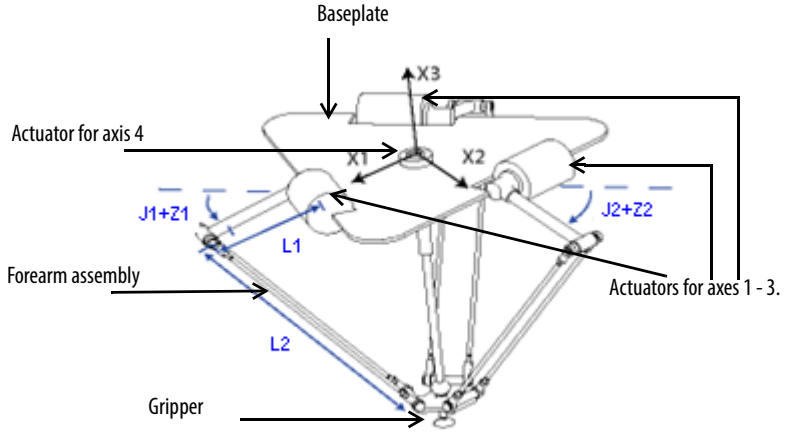
- Three-dimensional Delta
- Two-dimensional Delta
- SCARA Delta

In these geometries, the number of joints is greater than the degrees of freedom, and not all the joints are actuated (motor driven). These un-actuated joints are typically spherical joints.

Configure a Delta Three-dimensional Robot

This illustration shows a four axes Delta robot that moves in three-dimensional Cartesian (X1, X2, X3) space. This type of robot is often called a spider or umbrella robot.

Figure 17 - Delta Three-dimensional Robot



The Delta robot in this illustration is a three-degree of freedom robot with an optional fourth degree of freedom used to rotate a part at the tool tip. In Logix Designer application, the first three-degrees of freedom are configured as three joint axes (J1, J2, J3) in the robots coordinate system. The three joint axes are either:

- directly programmed in joint space.
- automatically controlled by the embedded Kinematics software in Logix Designer application from instructions programmed in a virtual Cartesian coordinate system.

This robot contains a fixed top plate and a moving bottom plate. The fixed top plate is attached to the moving bottom plate by three link-arm assemblies. All three of the link-arm assemblies are identical in that they each have a single top link arm (L1) and a parallelogram two-bar link assembly (L2).

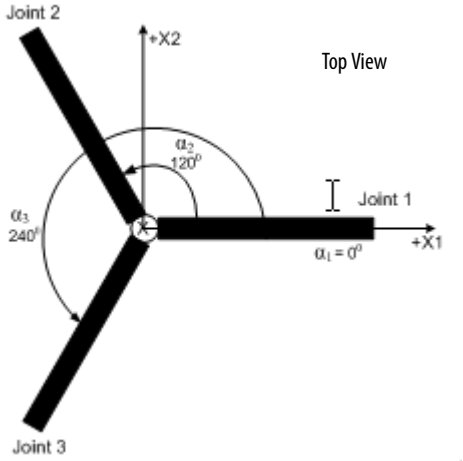
As each axis (J1, J2, J3) is rotated, the TCP of the gripper moves correspondingly in (X1, X2, X3) direction. The gripper remains vertical along the X3 axis while its position is translated to (X1, X2, X3) space by the mechanical action of the parallelograms in each of the three forearm assemblies. The mechanical connections of the parallelograms via spherical joints ensure that the top and bottom plates remain parallel to each other.

You program the TCP to an (X1, X2, X3) coordinate, then Logix Designer application computes the commands necessary for each of the joints (J1, J2, J3) to move the gripper linearly from the current (X1, X2, X3) position to the programmed (X1, X2, X3) position, at the programmed vector dynamics.

When each top link (L1) moves downward, its corresponding joint axis (J1, J2, or J3) is assumed to be rotating in the positive direction. The three joint axes of the robot are configured as linear axes.

To rotate the gripper, configure a fourth axis as either a linear or rotary, independent axis.

Establish the Reference Frame for a Delta Three-dimensional Robot



The reference frame for the Delta geometries is at the center of the top fixed plate. Joint 1, Joint 2, and Joint 3 are actuated joints. If you configure the Delta coordinate system in Logix Designer application with the joints homed at 0° in the horizontal position, then L1 of one of the link pairs is aligned along the X1 positive axis as shown. Moving in the counter clockwise direction from Joint 1 to Joint 2, the X2 axis is orthogonal to the X1 axis. Based on the right hand rule, X3 positive is the axis pointing up (out of the paper).

Calibrate a Delta Three-dimensional Robot

Use these steps to calibrate your robot.

1. Obtain the angle values from the robot manufacturer for J1, J2, and J3 at the calibration position. These values are used to establish the reference position.
2. Move all joints to the calibration position by either jogging the robot under programmed control, or manually moving the robot when the joint axes are in an open loop state.
3. Do **one** of these:
   a. Use a Motion Redefine Position instruction (MRP) to set the positions of the joint axes to the calibration values obtained in step 1.

b. Set the configuration value for the joint axes home position to the calibration values obtained in step 1 of this procedure and execute a Motion Axis Home instruction (MAH) for each joint axis.

4. Move each joint to an absolute position of 0.0. Verify that each joint position reads 0° and that the respective L1 is in a horizontal position. If L1 is not in a horizontal position, then see the alternate method for calibrating a Delta three-dimensional robot.

Alternate Method for Calibrating a Delta Three-dimensional Robot

Rotate each joint to a position so that the respective link is at a horizontal position, and then perform **one** of the following:

- Use a MRP instruction to set all the joint angles to 0° at this position.
- Configure the values for the Zero Angle Offsets to be equal to the values of the joints when in a horizontal position.

Configure Zero Angle Orientations for Delta Three-dimensional Robot

For Delta robot geometries, the internal transformation equations in the Logix Designer application are written assuming that:

- joints are at 0° when link L1 is horizontal.
- as each top link (L1) moves downward, its corresponding joint axis (J1, J2, or J3) is rotating in the positive direction.

If you want the joint angular position when L1 is horizontal to be at any other value than 0°, then you must properly configure the Zero Angle Orientation values on the Geometry tab of the Target Coordinate System Properties dialog to align your joint angle positions with the internal equations.

For example, if your Delta robot is mounted so that the joints attached at the top plate are homed at 30° in the positive direction below horizontal (see Delta Robot with Joints Homed at 30° illustration below) and you want the Logix Designer application readout values to be zero in this position, then you must configure the Zero Angle Orientation values to -30° on the Geometry tab of the Target Coordinate System Properties dialog (see the Configuring Delta Robot Zero Angle Orientation illustration below).

Figure 18 - Delta Robot with Joints Homed at 30°

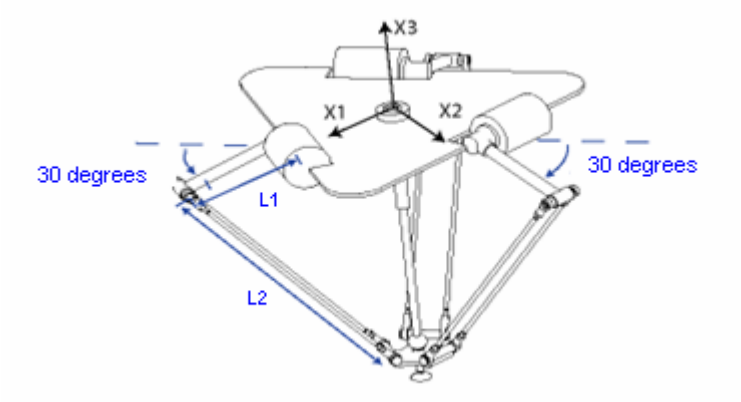
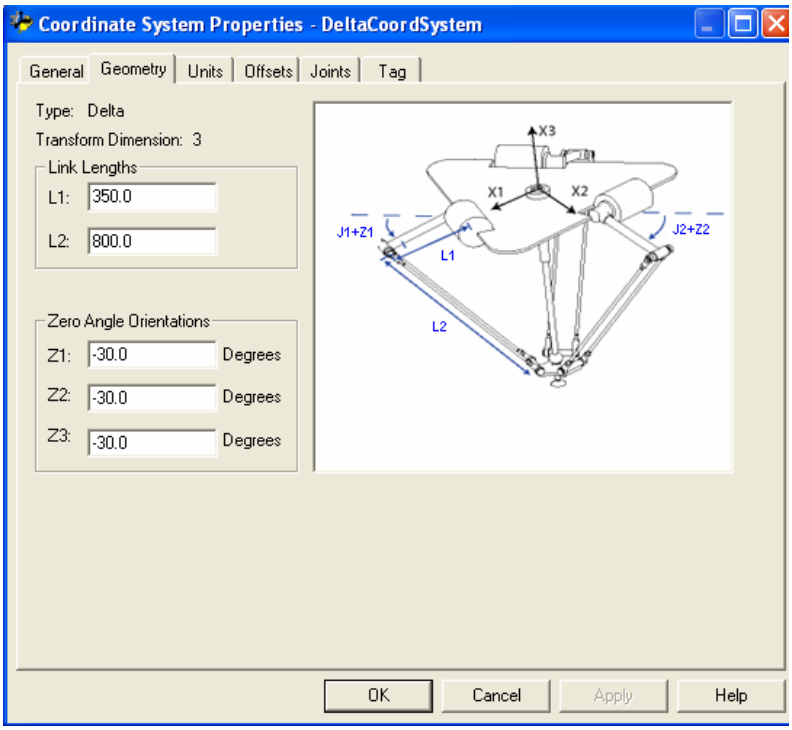


Figure 19 - Configuring Delta Robot Zero Angle Orientation



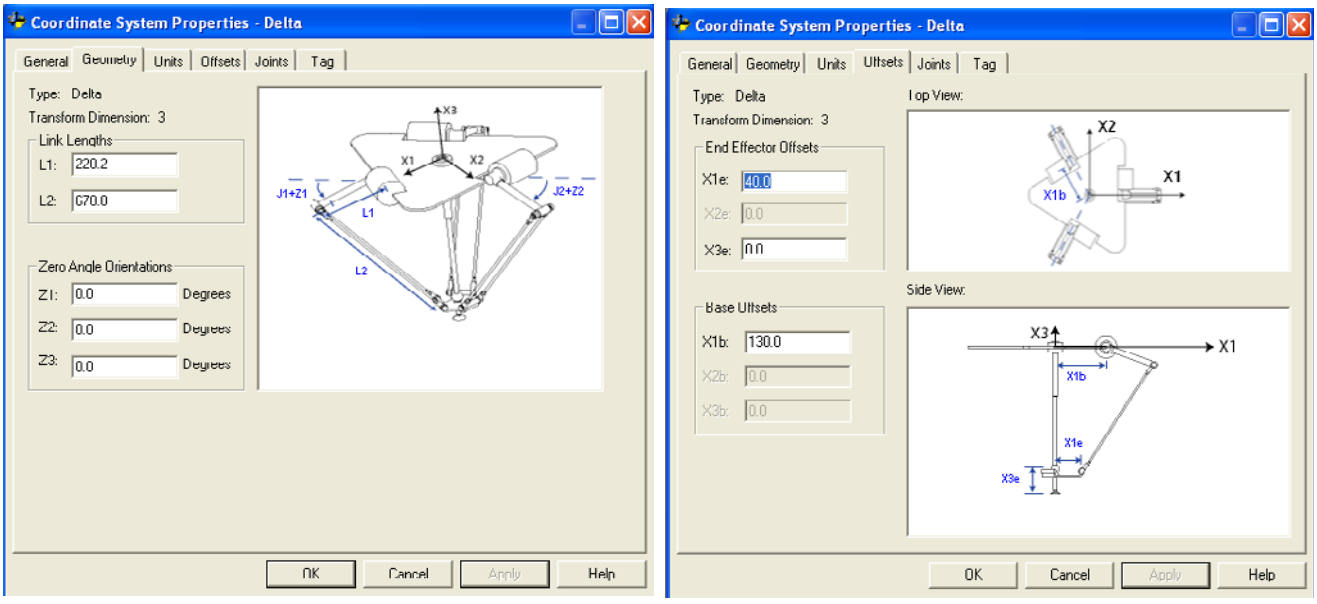
Identify the Work Envelope for a Delta Three-dimensional Robot

The work envelope is the three-dimensional region of space that defines the reaching boundaries for the robot arm. The typical work envelope for a Delta robot can be described as looking similar to plane in the upper region, with sides similar to a hexagonal prism and the lower portion similar to a sphere. For more detailed information regarding the work envelope of your Delta three-dimensional robot, see the documentation provided by the robot manufacturer.

We recommend that you program the robot within a rectangular solid defined inside the robots work zone. The rectangular solid can be defined by the positive and negative dimensions of the X1, X2, X3 virtual source axes. Be sure that the robot position does not go outside the rectangular solid. You can check the position in the event task.

To avoid issues with singularity positions, Logix Designer application internally calculates the joint limits for the Delta robot geometries. When an MCT instruction is invoked for the first time, the maximum positive and maximum negative joint limits are internally calculated based upon the link lengths and offset values entered on the Geometry and Offsets tabs of the Coordinate System Properties dialog.

Figure 20 - Delta Three-dimensional Configuration Systems Properties Dialog - Geometry and Offsets Tabs



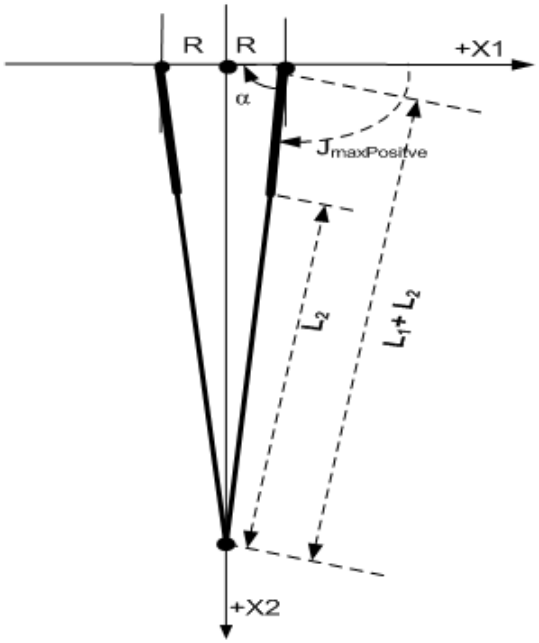
During each scan, Logix Designer application evaluates the joint positions in the forward and inverse kinematics routines to be sure that they do not violate the computed maximum positive and maximum negative joint limits.

Homing or moving a joint axis to a position beyond a computed joint limit and then invoking a MCT instruction, results in an error 67 (Invalid Transform position). For more information regarding error codes, See Error Codes (ERR) for Coordinate Motion Instructions on page 261.

Maximum Positive Joint Limit Condition

The derivations for the maximum positive joint apply to the condition when L1 and L2 are collinear.

Figure 21 - Maximum Positive Joint Limit Condition - L1 and L2 are Collinear



Maximum Positive Joint Limit Position
R = absolute value of (X1b - X1e)

$$\alpha = \cos^{-1}\left(\frac{R}{L1 + L2}\right)$$

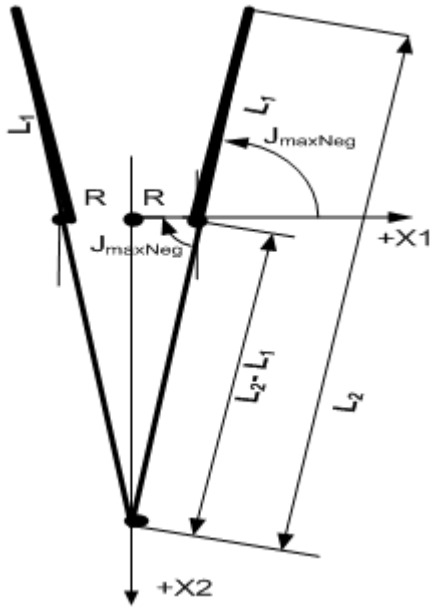
$$\text{Jmax Positive} = 180^{\circ} - \alpha$$

Maximum Negative Joint Limit Condition

The derivations for the maximum negative joint limit applies to the condition when L1 and L2 are folded back on top of each other.

R is computed by using the base and end-effector offsets values (X1b and X1e).

Figure 22 - Maximum Negative Joint Limit Condition - L1 and L2 are Folded Back on Top of Each Other



Maximum Negative Joint Limit Condition
R = absolute value of (X1b - X1e)

$$JMaxNeg = -\cos^{-1}\left(\frac{R}{L2 - L1}\right)$$

Define Configuration Parameters for a Delta Three-dimensional Robot

Logix Designer application can be configured for control of robots with varying reach and payload capacities. As a result, it is very important to know the configuration parameter values for your robot including:

- Link lengths.
- Base offsets.
- End-effector offsets.

The configuration information is available from the robot manufacturer.

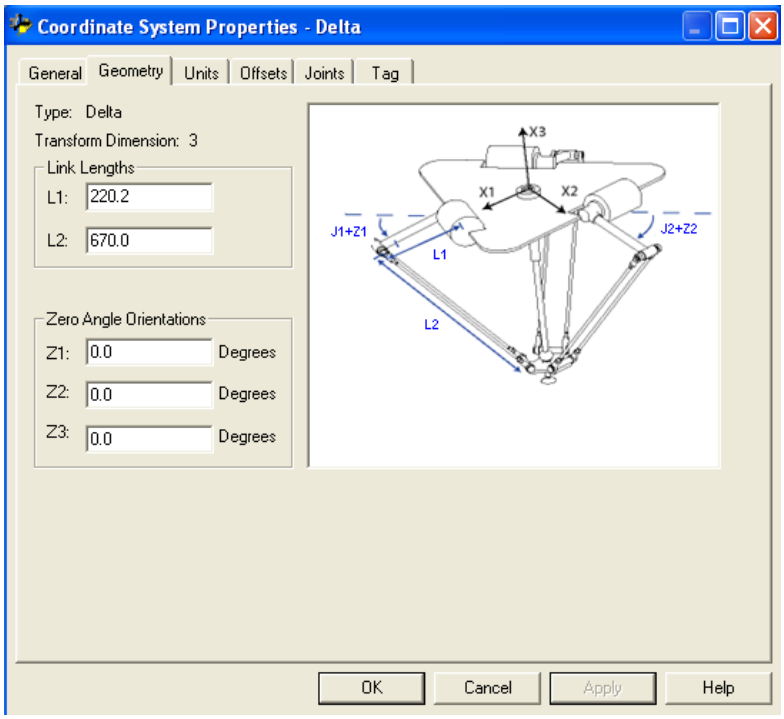
IMPORTANT	Verify that the values for the link lengths, base offsets, and end-effector offsets are entered into the Configuration Parameters dialog by using the same measurement units.

Link Lengths

Link lengths are the rigid mechanical bodies attached at the rotational joints. The three-dimensional Delta robot geometry has three link pairs each made up of L1 and L2. Each of the link pairs has the same dimensions.

- L1 - is the link attached to each actuated joint (J1, J2, and J3).
- L2 - is the parallel bar assembly attached to L1.

Figure 23 - Three-dimensional Delta Robot - Link Lengths Configuration Screen



Base Offsets

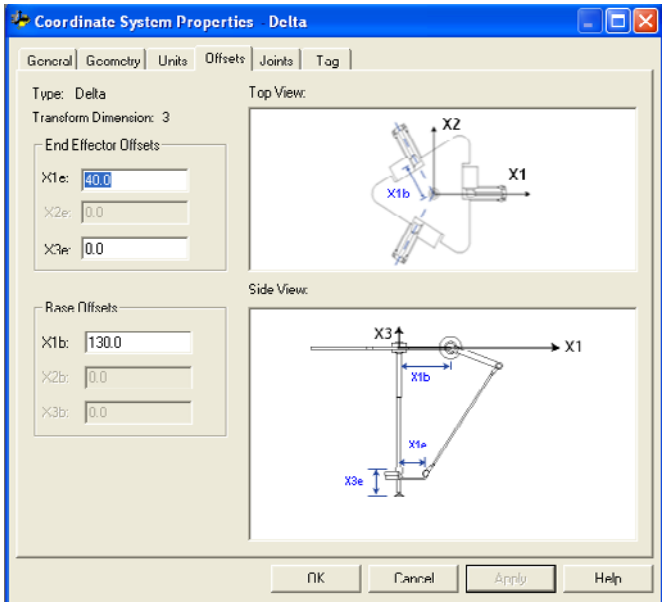
There is one base offset value (X1b) available for the three-dimensional Delta robot geometry. Enter the value equal to the distance from the origin of the robot coordinate system to one of the actuator joints.

End-effector Offsets

The two end-effector offsets available for the three-dimensional Delta robot geometry are as follows. Offset values are always positive numbers.

- X1e is the distance from the center of the moving plate to the lower spherical joints of the parallel arms.
- X3e is the distance from the base plate to the TCP of the gripper.

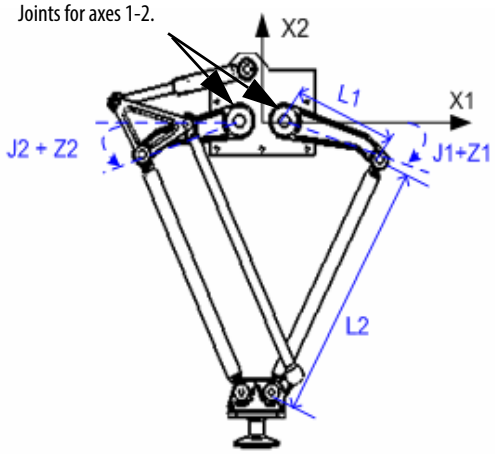
Figure 24 - Configuring the Base Offset and End-effector Offsets for a Three-dimensional Delta Robot



Configure a Delta Two-dimensional Robot

This illustration shows a two-dimensional Delta robot that moves in two-dimensional Cartesian space.

Figure 25 - Two-dimensional Delta Robot



This robot has two rotary joints that move the gripper in the (X1, X2) plane. Two forearm assemblies attach a fixed top plate to a movable bottom plate. A gripper is attached to the movable bottom plate. The bottom plate is always orthogonal to the X2 axis and its position is translated in Cartesian space (X1, X2) by mechanical parallelograms in each forearm assembly. The two joints, J1, and J2, are actuated joints. The joints between links L1 and L2 and between L2 and the base plate are unactuated joints.

Each joint is rotated independently to move the gripper to a programmed (X1, X2) position. As each joint axis (J1 or J2 or J1 and J2) is rotated, the TCP of the gripper moves correspondingly in the X1 or X2 direction or X1 and X2 direction. You can program the TCP to a (X1, X2) coordinate, then Logix Designer application uses internal vector dynamic calculations to compute the proper commands needed for each joint to move the gripper linearly from the current (X1, X2) position to the programmed (X1, X2) position.

The two joint axes (J1 and J2) of the robot are configured as linear axes.

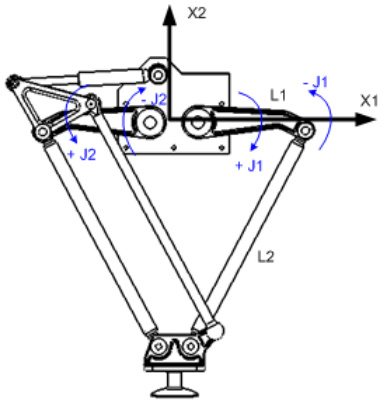
To rotate the gripper, configure a third axis as a linear or rotary, independent axis.

Establish the Reference Frame for a Delta Two-dimensional Robot

The reference frame for the two-dimensional Delta geometry is at the center of the fixed top plate. When the angles of joints J1 and J2 are both at 0°, each of the two L1 links is along the X1 axis. One L1 link is pointing in the positive X1 direction, the other in the negative X1 direction.

When the right-hand link L1 moves downward, joint J1 is assumed to be rotating in the positive direction and when L1 moves upward, the J1 is assumed to be moving in the negative direction. When the left-hand link L1 moves downward, joint J2 is assumed to be rotating in the positive direction and when left-hand L1 moves upward, the J2 is assumed to be moving in the negative direction.

Figure 26 - Establishing the Two-dimensional Delta Robot Reference Frame



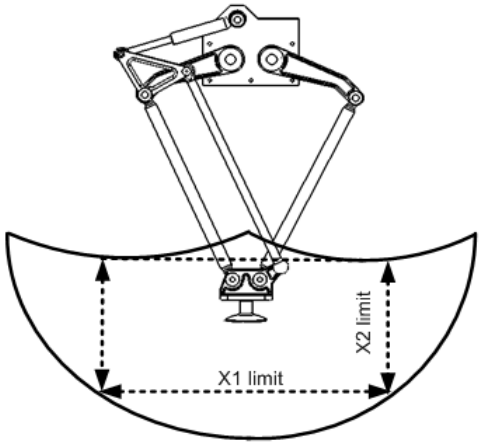
Calibrate a Delta Two-dimensional Robot

The method used to calibrate a Delta two-dimensional robot is the same as the method used for calibrating a Delta three-dimensional robot. The only difference is the number of axes used. For more information about calibration, see Calibrate a Delta Three-dimensional Robot on page 66.

Identify the Work Envelope for a Delta Two-dimensional Robot

The work envelope is the two-dimensional region of space that defines the reaching boundaries for the robot arm. The typical working envelope for a two-dimensional Delta robot is a boundary composed of circular arcs.

Figure 27 - Work Envelope for Two-dimensional Delta Robot



We recommend that you define the program parameters for the two-dimensional Delta robot within a rectangle (dotted lines in the figure above) inside the robots work zone. The rectangle can be defined by the positive and negative dimensions of the X1, X2 virtual source axes. Be sure that the robot position does not go outside the rectangle. You can check the position in the event task.

To avoid problems with singularity positions, Logix Designer application internally calculates the joint limits for the Delta robot geometries. When an MCT instruction is invoked for the first time, the maximum positive and maximum negative joint limits are internally calculated based upon the link lengths and offset values entered on the Geometry and Offsets tabs of the Coordinate System Properties dialog

For More Information About	Page
Maximum positive joint limits	70
Maximum negative joint limits	71

Homing or moving a joint axis to a position beyond a computed joint limit and then invoking an MCT instruction, results in an error 67 (Invalid Transform position). See Error Codes (ERR) for Coordinate Motion Instructions on page 261 for more information regarding error codes.

Define Configuration Parameters for a Delta Two-dimensional Robot

You can configure Logix Designer application for control of robots with varying reach and payload capacities. As a result, it is very important to know the configuration parameter values for your robot including:

- Link lengths.
- Base offsets.
- End-effector offsets.

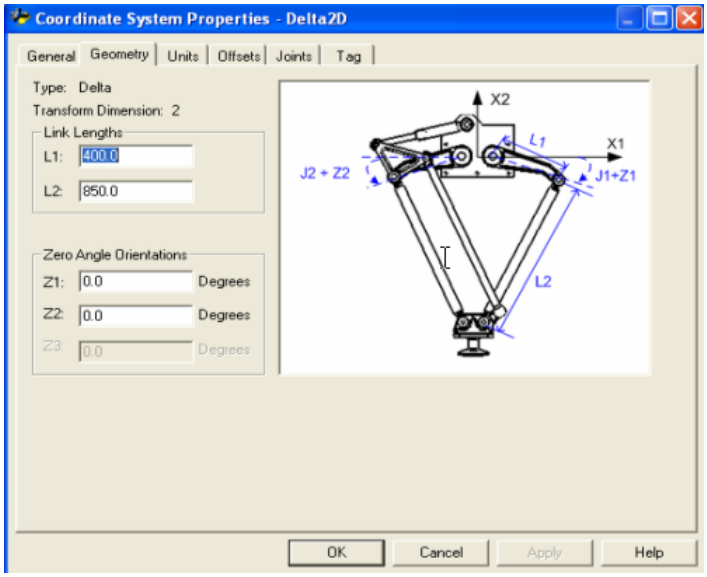
The configuration information is available from the robot manufacturer.

IMPORTANT	Verify that the values for the link lengths, base offsets, and end-effector offsets are entered into the Configuration Parameters dialog by using the same measurement units.

Link Lengths

Links are the rigid mechanical bodies attached at joints. The two- dimensional Delta geometry has two link pairs, each with the same lengths. The link attached to each actuated joint (J1 and J2) is L1. The parallel bar assembly attached to link L1 is link L2.

Figure 28 - Two-dimensional Delta Robot - Link Lengths Configuration Screen



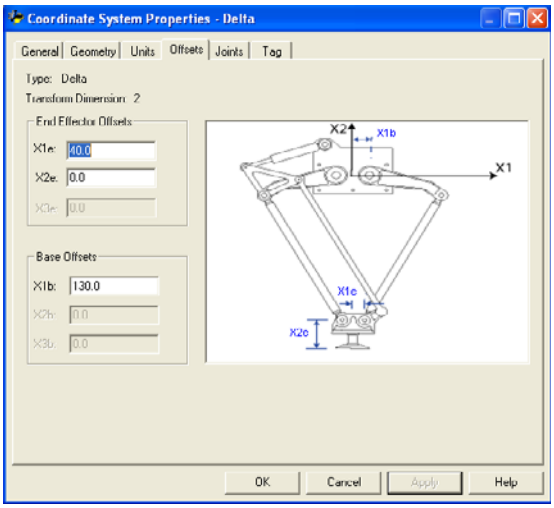
Base Offsets

There is one base offset (X1b) available for the two-dimensional Delta robot geometry. Enter the value equal to the distance from the origin of the robot coordinate system to one of the actuator joints.

End-effector Offsets

There are two end-effector offsets available for the two-dimensional Delta robot geometry. The value for X1e is the offset distance from the center of the lower plate to the lower spherical joints of the parallel arms. The distance from the lower plate to the TCP of the gripper is the value for X2e.

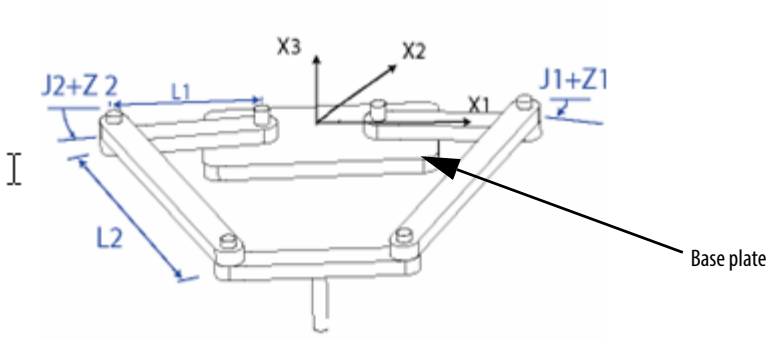
Figure 29 - Delta Two-dimensional Robot - Base and End-effector Offsets



Configure a SCARA Delta Robot

The SCARA Delta robot geometry is similar to a two-dimensional Delta robot geometry except that the X1-X2 plane is tilted horizontally with the third linear axis in the vertical direction (X3).

Figure 30 - SCARA Delta Robot



Establish the Reference Frame for a SCARA Delta Robot

The reference frame for the SCARA Delta robot is at the center of the base plate.

When the angles of joints J1 and J2 are both at 0°, each of the two L1 links is along the X1 axis. One L1 link is pointing in the positive X1 direction, the other in the negative X1 direction.

When the right-hand link L1 moves in the clockwise direction (looking down on the robot), joint J1 is assumed to be rotating in the positive direction. When the right-hand link L1 moves counterclockwise, joint J1 is assumed to be moving in the negative direction.

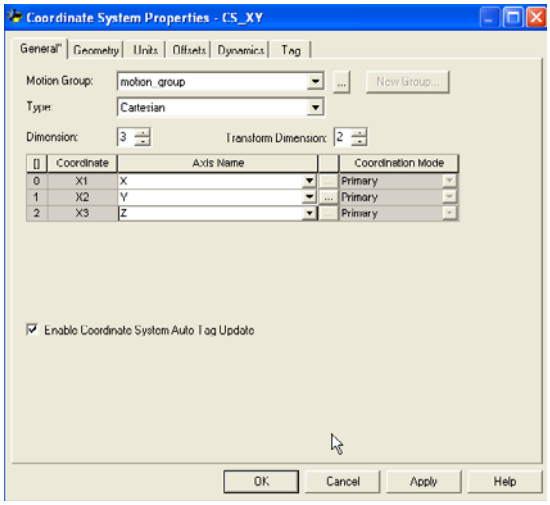
When left-hand link L1 moves in the clockwise direction, joint J2 is assumed to be moving in the negative direction. When the left-hand link L1 moves in the counterclockwise direction, joint J2 is assumed to be rotating in the positive direction.

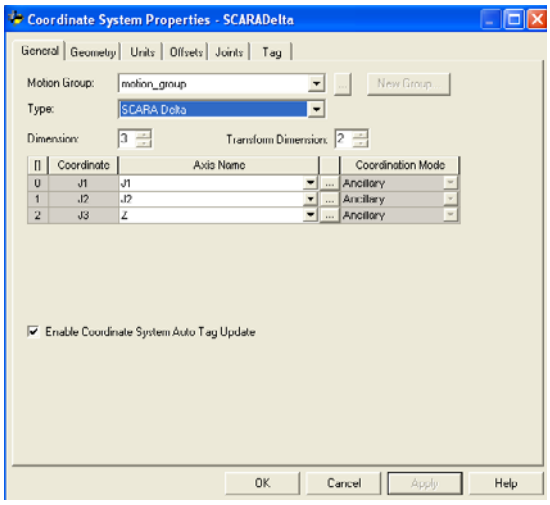
Based on the right hand rule, X3 positive is orthogonal to the X1-X2 plane pointing up. The linear axis always moves in the X3 direction.

When configuring a SCARA Delta robot in Logix Designer application, keep the following in mind.

- Configure **both** the source and the target coordinate system with a transform dimension of two.
- The linear axis configured as a third axis must be the same for both the source and target coordinate systems.

Figure 31 - Example of Source and Target Coordinate System Configuration for a SCARA Delta Robot





Calibrate a SCARA Delta Robot

The method used to calibrate a SCARA Delta robot is the same as the method used for calibrating a Delta three-dimensional robot. The only difference is the number of axes used. For more information about calibration, see Calibrate a Delta Three-dimensional Robot on page 66.

Identify the Work Envelope for a SCARA Delta Robot

The work envelope for a SCARA Delta robot is similar to the two-dimensional Delta robot in the X1-X2 plane. The third linear axis extends the work region making it a solid region. The maximum positive and negative limits of the linear axis defines the height of the solid region.

We recommend that you program the SCARA Delta robot within a rectangular solid defined inside the robots work zone. The rectangular solid can be defined by the positive and negative dimensions of the X1, X2, X3 virtual source axes. Be sure that the robot position does not go outside the rectangular solid. You can check the position in the event task.

To avoid problems with singularity positions, Logix Designer application internally calculates the joint limits for the Delta robot geometries.

For More Information About	Page
Maximum positive joint limits	70
Maximum negative joint limits	71

Homing or moving a joint axis to a position beyond a computed joint limit and then invoking an MCT instruction, results in an error 67 (Invalid Transform position). See Error Codes (ERR) for Coordinate Motion Instructions on page 261 for more information regarding error codes.

Define Configuration Parameters for a SCARA Delta Robot

Logix Designer application can be configured for control of robots with varying reach and payload capacities. As a result, it is very important to know the configuration parameter values for your robot including:

- Link lengths.
- Base offset.
- End-effector offset.

The configuration information is available from the robot manufacturer.

IMPORTANT	Verify that the values for the link lengths, base offsets, and end-effector offsets are entered into the Configuration Parameters dialog by using the same measurement units.

Link Lengths

Links are the rigid mechanical bodies attached at joints. The SCARA Delta robot has two link pairs each with the same lengths. The link attached to each actuated joint (J1 and J2) is L1. The parallel bar assembly attached to link L1 is link L2.

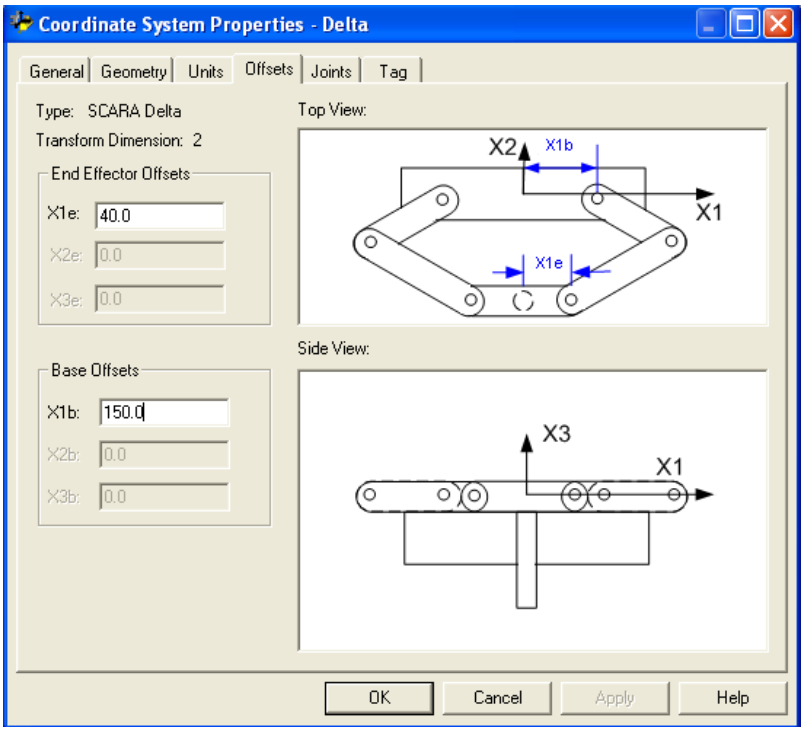
Base Offset

There is one base offset (X1b) available for the SCARA Delta robot geometry. Enter the value equal to the distance from the origin of the robot coordinate system to one of the actuator joints. The base offset value is always a positive number.

End-effector Offset

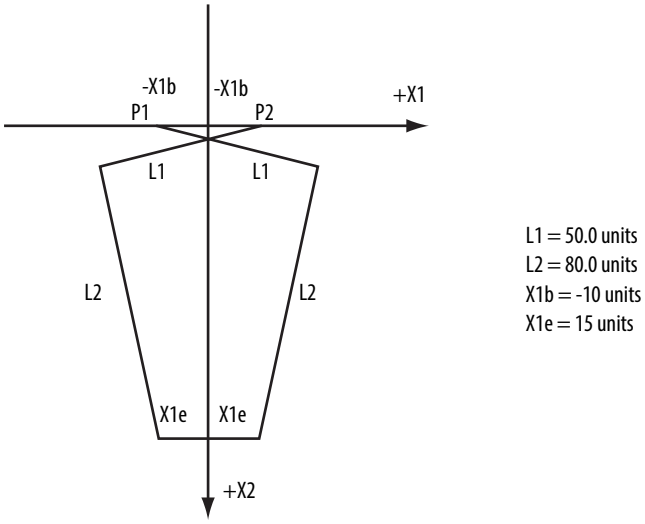
There is one end-effector offset (X1e) available for the SCARA Delta robot geometry. Enter the value for the distance from the center of the moving plate to one of the spherical joints of the parallel arms. The end-effector value is always a positive number.

Figure 32 - SCARA Delta End-effector and Base Offset



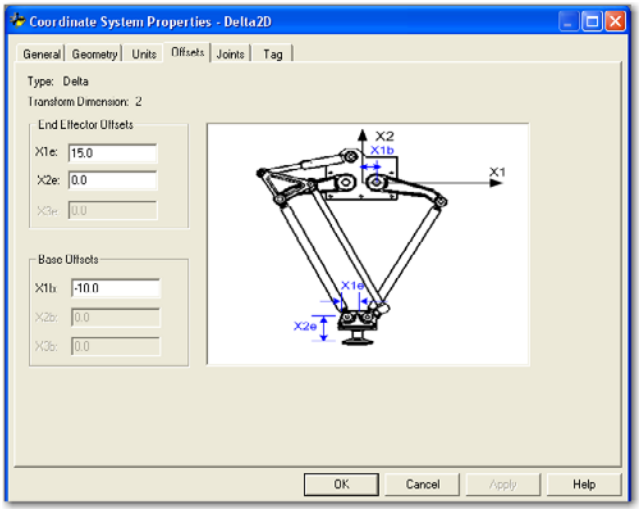
Configure a Delta Robot with a Negative X1b Offset

Beginning with version 17 of the application, you can use negative offsets for the X1b base offset on both 2D and 3D delta geometries. For example, a mechanical 2D delta robot that uses a negative X1b offset has a mechanical configuration like the one shown below.



The base offset X1b is the value equal to the distance from the origin of the robot coordinate system to one of the actuator joints. In the previous figure, one of the actuator joints (P1), is on the negative side of X1. Therefore, the base offset X1b is measured to be a value of -10 units from the origin of the coordinate system (X1 - X2 intersection) to P1.

The Logix Designer application coordinate system configuration for the offset tab used with the example above is shown below.



This negative offset description also applies for Delta 3D and SCARA-Delta Configurations.

Arm Solutions

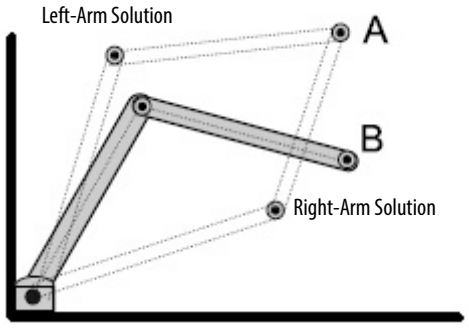
A Kinematic arm solution is the position of all joints on the robot that correspond to a Cartesian position. When the Cartesian position is inside the workspace of the robot, then at least one solution always exists. Many of the geometries have multiple joint solutions for a single Cartesian position.

- Two axis robots - two joint solutions typically exist for a Cartesian position.
- Three axis robots - four joint solutions typically exist for a Cartesian position.

Left-Arm and Right-Arm Solutions for Two-Axes Robots

A robot having an arm configuration has two Kinematics solutions when attempting to reach a given position (point A shown on the figure below). One solution satisfies the equations for a right-armed robot, the other solution satisfies the equations for a left-armed robot.

Figure 33 - Right Arm and Left Arm Solutions



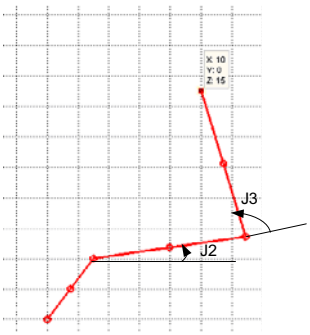
Solution Mirroring for Three-dimensional Robots

For a three-dimensional Articulated Independent robot, there are four possible solutions for the same point.

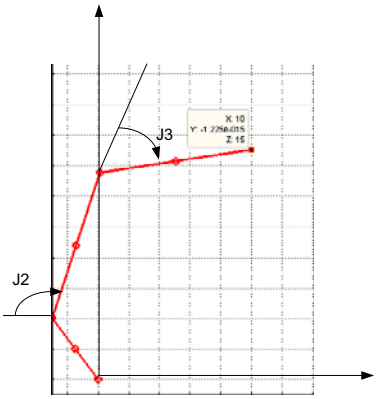
- Left-Arm
- Right-Arm
- Left-Arm Mirror
- Right-Arm Mirror

For example, consider the Cartesian point XYZ (10,0,15). The joint position corresponding to this point has four joint solutions. Two of the solutions are the same as the solutions for the two-dimensional case. The other two solutions are mirror image solutions where J1 is rotated 180°.

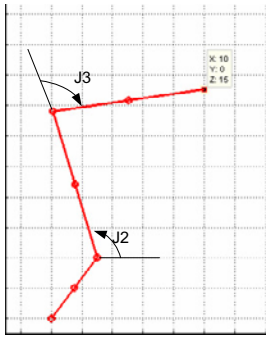
Right-Arm



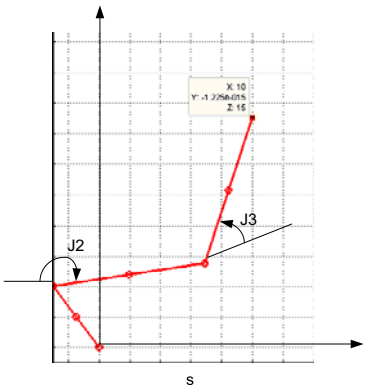
Right-Arm Mirror



Left-Arm



Left-Arm Mirror



Activating Kinematics

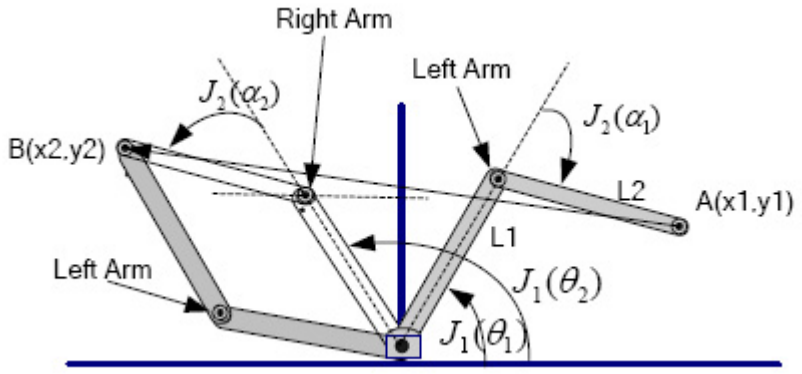
WARNING: Be sure to choose an arm solution before activating the Kinematic function. Failure to do so can result in machine damage and/or serious injury or death to personnel.

Before activating Kinematics, configure the robot in a left-arm or right-arm solution. The robot stays in the same configuration in which it was activated as it is moved in Cartesian or source coordinate mode. If activated in a fully-extended-arm mode (this is, neither a left-arm nor a right-arm solution), the system chooses a left-arm solution.

Change the Robot Arm Solution

You can switch the robot from a left-arm solution to a right-arm solution or vice versa. This is done automatically when a joint move is programmed forcing a left/right change to occur. After the change is performed, the robot stays in the new arm solution when Cartesian moves are made. The robot arm solution changes again (if required) when another joint move is made.

Example: Suppose, you want to move the robot from position A (x1,y1) to position B (X_2,Y_2) (see the figure below). At position A, the system is in a left arm solution. Programming a Cartesian move from A (X_1,Y_1) to B (X_2,Y_2) means that the system moves along the straight line (see the illustration) from A to B while maintaining a left arm solution. If you want to be at position B in a right-arm solution, you must make a joint move in J1 from θ_1 to θ_2 and a joint move in J2 from α_1 to α_2.



Plan for Singularity

A singularity occurs when an infinite number of joint positions (mathematical solutions) exist for a given Cartesian position. The Cartesian position of a singularity is dependent on the type of the robot geometry and the size of the link lengths for the robot. Not all robot geometries have singularity positions.

For example, singularities for an Articulated Independent robot occur when:

- the robot manipulator folds its arm back onto itself and the Cartesian position is at the origin.
- the robot is fully stretched at or very near the boundary of its workspace.

An error condition is generated when a singularity position is reached.

WARNING: Avoid programming your robot towards a singularity position when programming in Cartesian mode. The velocity of the robot increases very rapidly as it approaches a singularity position and can result in injury or death to personnel.

Encounter a No-solution Position

WARNING: Avoid programming your robot towards a no solution position when programming in Cartesian mode. The velocity of the robot increases very rapidly as it approaches this position and can result in injury or death to personnel.

When a robot is programmed to move beyond its work envelope, there is no mathematical joint position for the programmed Cartesian position. The system forces an error condition.

For example, if an Articulated Independent robot has two 10-inch arms, the maximum reach is 20 inches. Programming to a Cartesian position beyond 20 inches produces a condition where no mathematical joint position exists.

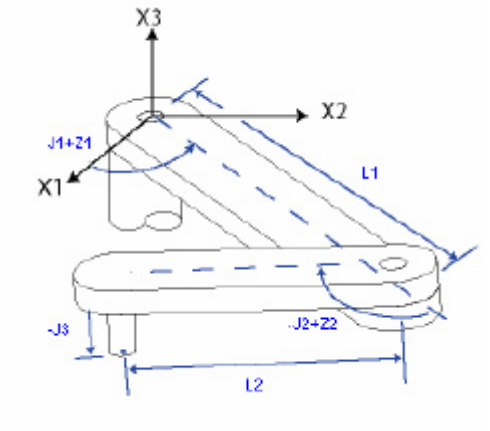
Configure a SCARA Independent Robot

The typical SCARA Independent robot has two revolute joints and a single prismatic joint. This robot is identical to the Articulated Independent two-dimensional robot except that the X1-X2 plane is tilted horizontally with a third linear axis in the vertical direction. Use these guidelines when configuring a SCARA Independent robot.

Establish the Reference Frame for a SCARA Independent Robot

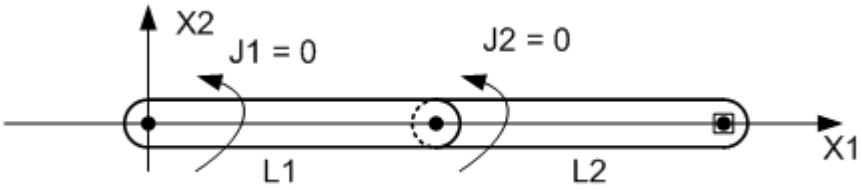
The reference frame for the SCARA Independent geometry is at the base of link L1.

Figure 34 - SCARA Independent Robot Reference Frame



The internal Kinematic equations are written as if the start position for the SCARA Independent robot joints are as shown in this illustration.

Figure 35 - Joint and Link Start Position that Kinematics Equations use for the SCARA Independent Robots



Top View

- +J1 is measured counterclockwise around +X3 axis starting at an angle of J1 =0.0 when L1 is along the X1 axis.
- +J2 is measured counterclockwise starting with J2 = 0 when Link L2 is aligned with Link L1.
- +J3 is a prismatic axis that moves parallel to +X3 axis.

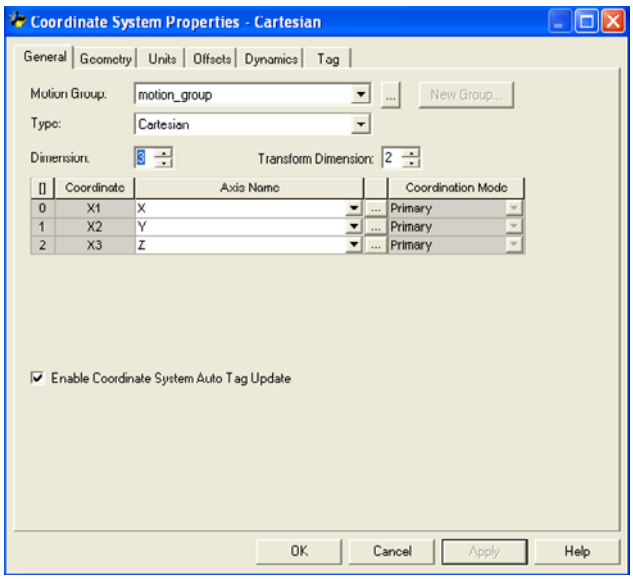
For information about alternate methods for establishing a reference frame, see Articulated Independent Robot on page 55.

When configuring the parameters for the Source coordinate system and the Target coordinate system for a SCARA Independent robot, keep the following information in mind:

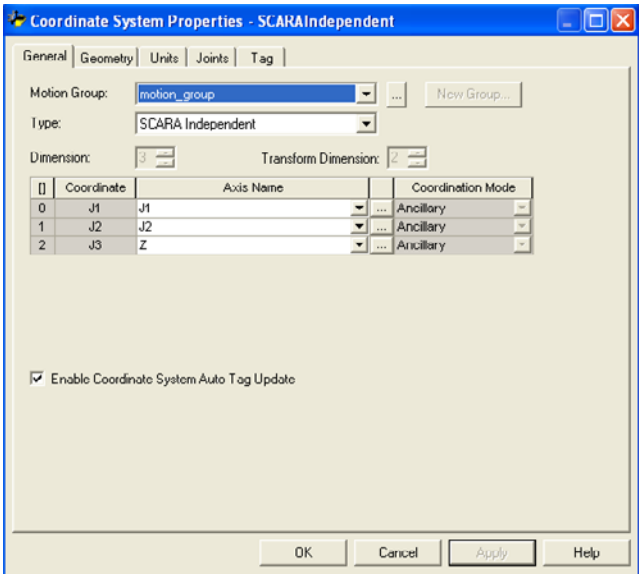
- Set the transform dimension value to two for both the source and target coordinate systems because only J1 and J2 are involved in the transformations.
- The Z axis is configured as a member of both the source and target coordinate systems.

For additional information about establishing a reference frame, see Articulated Independent Robot on page 55.

Figure 36 - Example Source and Target Coordinate Systems for a SCARA Independent Robot



Source Coordinate System Configuration



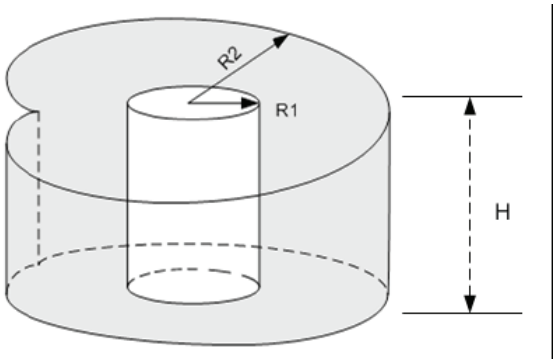
Target Coordinate System Configuration

Identify the Work Envelope for a SCARA Independent Robot

The work envelope is the three-dimensional region of space that defines the reaching boundaries for the robot arm. The work envelope for the SCARA Independent robot is a hollow cylinder with:

- a height equal to the travel limit of the J3 axis.
- an inner radius (R1) equal to |L1-L2|.
- an outer radius (R2) equal to |L1+L2|.

Figure 37 - Example Work Envelope for a SCARA Independent Robot



Define Configuration Parameters for a SCARA Independent Robot

Logix Designer application can be configured for control of robots with varying reach and payload capacities. As a result, it is very important to know the configuration parameter values for your robot including:

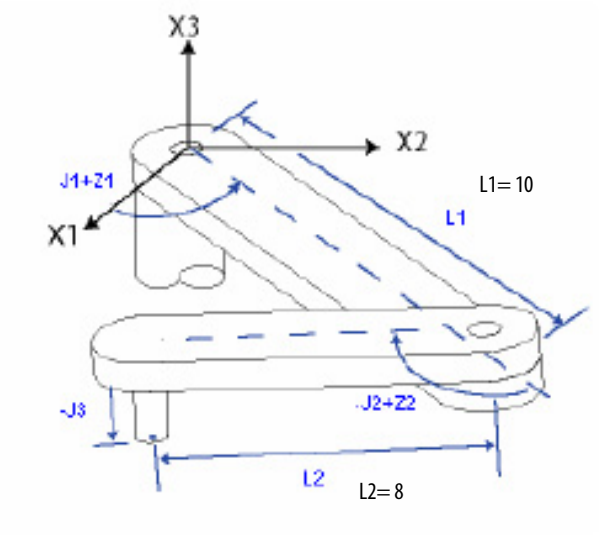
- Link lengths.
- Base offsets.
- End-effector offsets.

The configuration information is available from the robot manufacturer.

IMPORTANT	Verify that the values for the link lengths, base offsets, and end-effector offsets are entered into the Configuration Parameters dialog by using the same measurement units.

The following example illustrates the typical configuration parameters for a SCARA Independent robot.

Figure 38 - SCARA Independent



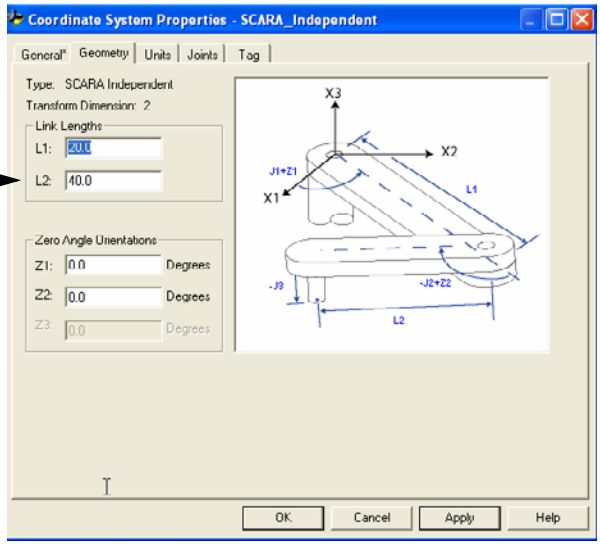
Link Lengths

Link lengths are the rigid mechanical bodies attached at joints.

Figure 39 - Configuring Link Lengths for a SCARA Independent Robot

Enter the Link Length values.
For the robot shown in SCARA Independent above, the Link Length values are:

- L1 = 20
- L2 = 40



Base offsets and end-effector offsets do not apply to a SCARA Independent robot configuration.

Error Conditions

Kinematics error conditions are detected:

- upon activation of a transformation by executing an MCT instruction.
- in some movement conditions.

  Errors can occur for certain movement conditions for either the source or target coordinate system after a transformation has been established. These types of errors are reported in the MCT instruction error codes. Singularity and other movement error conditions are also reported in the MCT error codes.
- computing an invalid position via an MCTP instruction.

See Error Codes (ERR) for Coordinate Motion Instructions on page 261 for more information regarding error codes.

Monitor Status Bits for Kinematics

You can monitor the status of the Kinematics functions by using Logix Designer application status bits.

To see if	Check this tag	And this bit	For
A coordinate system is the source of an active transform	Coordinate system	TransformSourceStatus	On
A coordinate system is the target of an active transform	Coordinate system	TransformTargetStatus	On
An axis is part of an active transform	Axis	TransformStateStatus	On
An axis is moving because of a transform	Axis	ControlledByTransformStatus	On

Notes:

Articulated Dependent Robot

Topic	Page
Reference Frame	93
Methods to Establish a Reference Frame	96
Work Envelope	98
Configuration Parameters	99

The Articulated Dependent robot has motors for the elbow and the shoulder at the base of the robot. The dependent link controls J3 at the elbow. Use these guidelines when configuring an Articulated Dependent robot.

WARNING: Before turning ON the Transform and/or establishing the reference frame, do the following for the joints of the target coordinate system:

WARNING: Set and enable the soft travel limits.

ATTENTION: Enable the hard travel limits.

WARNING: Failure to do this can allow the robot to move outside of the work envelope causing machine damage and/or serious injury or death to personnel.

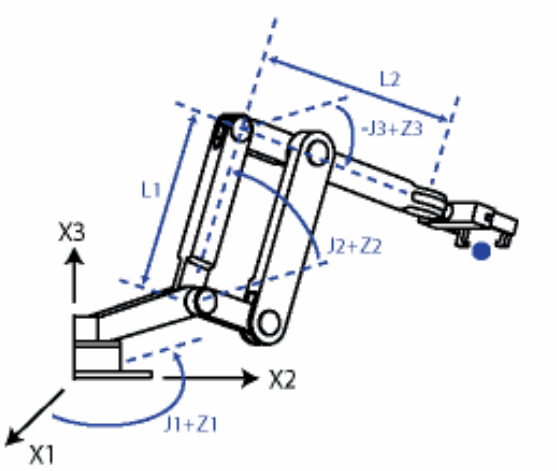
Reference Frame

The reference frame is the Cartesian (typically the source) coordinate frame that defines the origin and the three primary axes (X1, X2 and X3). These are used to measure the real Cartesian positions.

WARNING: Failure to properly establish the correct reference frame for your robot can cause the robotic arm to move to unexpected positions causing machine damage and/or injury or death to personnel.

The reference frame for an Articulated Dependent robot is at the base of the robot as shown in this figure.

Figure 40 - Articulated Dependent 1



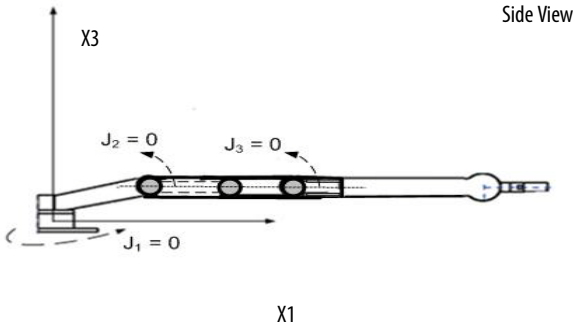
Before you begin establishing the Joint-to-Cartesian reference frame relationship, it is important to know some information about how the Kinematic mathematical equations in the ControlLogix controllers were written. The equations were written as if the Articulated Dependent robot joints were positioned as shown in Articulated Dependent 1.

- +J1 is measured counterclockwise around the +X3 axis starting at an angle of J1=0 when L1 and L2 are both in the X1-X2 plane.
- +J2 is measured counterclockwise starting with J2=0 when L1 is parallel to X1-X2 plane.
- +J3 is measured counterclockwise with J3=0 when L2 is parallel to the X1-X2 plane.

When your robot is physically in this position, the Logix Designer application Actual Position tags for the axes must be:

- J1 = 0.
- J2 = 0.
- J3 = 0.

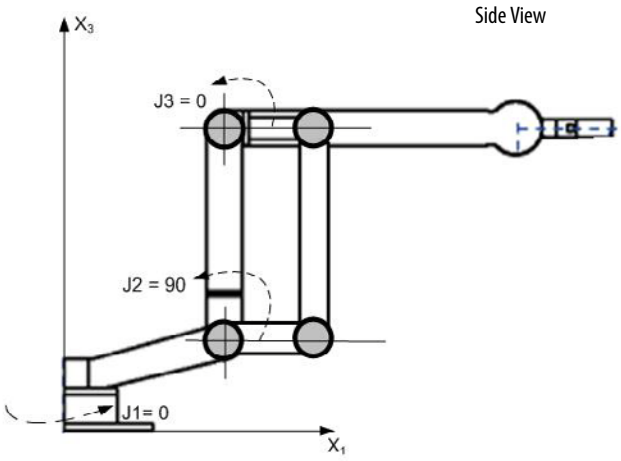
Figure 41 - Articulated Dependent 2



When your robot is physically in this position, the Logix Designer application Actual Position tags for the axes must be:

- J1 = 0.
- J2 = 90.
- J3 = -90.

Figure 42 - Articulated Dependent 3



If your robot's physical position **and** joint angle values cannot match those shown in Articulated Dependent 2 or in Articulated Dependent 3 then, use one of the methods outlined in this section to establish the Joint-to-Cartesian reference frame relationship.

WARNING: Failure to properly establish the correct reference frame for your robot can cause the robotic arm to move to unexpected positions potentially resulting in damage to property or injury to personnel.

Methods to Establish a Reference Frame

The following methods let you establish a reference frame for an Articulated Independent robot.

For each	Use one of these methods to establish the reference frame
Incremental axis	Each time the robot's power is cycled.
Absolute axis	Only when you establish absolute home.

- Method 1 - establishes a Zero Angle Orientation and lets the configured travel limits and home position on the joint axes remain operational. Use this method if you are operating the axes between the travel limits determined prior to programming a Motion Redefine Position (MRP) instruction and want these travel limits to stay operational.
- Method 2 - uses a Motion Redefine Position (MRP) instruction to redefine the axes position to align with the Joint reference frame. This method can require the soft travel limits to be adjusted to the new reference frame.

Method 1 - Establishing a Reference Frame

Each axis for the robot has the mechanical hard stop in each of the positive and negative directions. Manually move or press each axes of the robot against its associated mechanical hard stop and redefine it to the hard limit actual position provided by the robot manufacturer. J1 is the axis at the base of the robot that rotates around X3.

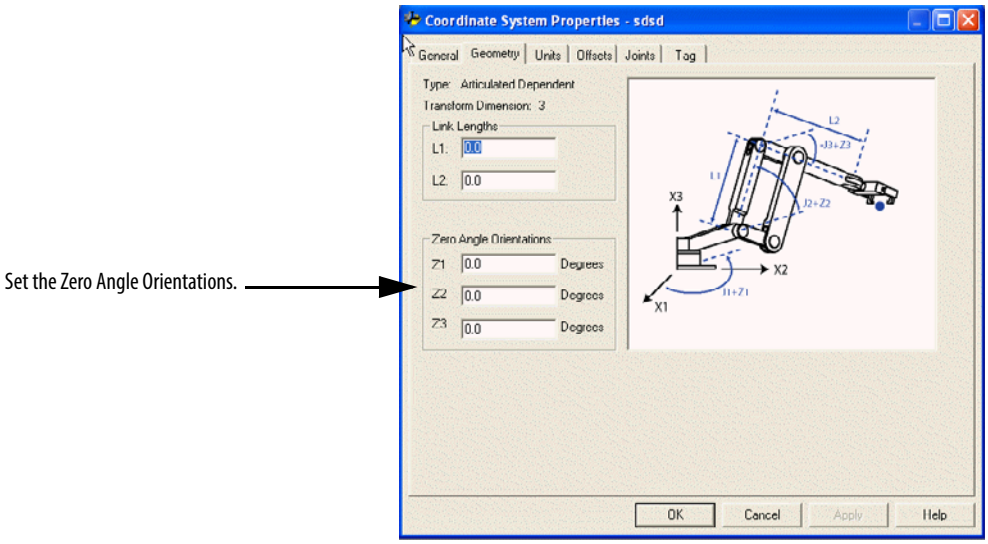
When the robot is moved so that Link1 is parallel to the X3 axis and Link2 is parallel to X1 axis as shown in Articulated Dependent 3, the Logix Designer application values for the Actual Position tags are:

- J1 = 0.
- J2 = 90°.
- J3 = 0°.

If the Logix Designer application Actual Position tags do not show these values, configure the Zero Angle Orientation for the joint or joints that do not correspond.

If the Logix Designer application read-out values are	Set the Zero Angle Orientations on the Coordinate System Properties dialog to
J1 = 10 J2 = 80 J3 = 5	Z1 = -10 Z2 = 10 Z3 = -5

Figure 43 - Example of Zero Angle Orientation for an Articulated Dependent Robot



Method 2 - Establishing a Reference Frame

Position the robot so that:

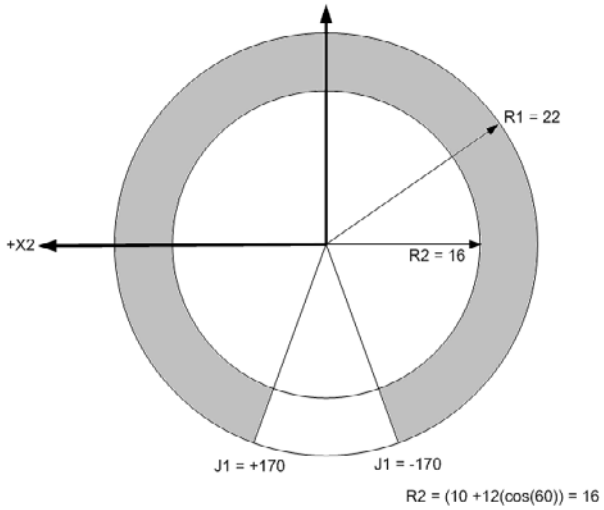
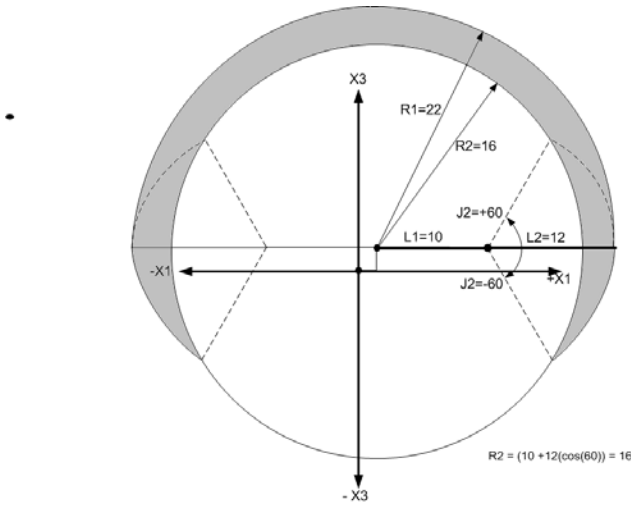
- L1 is parallel to the X3 axis.
- L2 is parallel to X1 axis.

Program a Motion Redefine Position (MRP) instruction for all the three axis to with the following values 0, 90, and 0°.

The Joint-to-Cartesian reference frame relationship is automatically established by the controller after the Joint coordinate system parameters (link lengths, base offsets, and end-effector offsets) are configured and the MCT instruction is enabled.

Work Envelope

The work envelope is the three-dimensional region of space defining the reaching boundaries for the robot arm. The work envelope of an articulated robot is ideally a complete sphere having an inner radius equal to |L1- L2| and outer radius equal to |L1+L2|. However, due to the range of motion limitations on individual joints, the work envelope is not always a complete sphere.

If the range-of-motion values for the articulated robot are	Typically, the work envelope is
J1 = ± 170 J2 = 0 to 180 J3 = ± 60 L1 = 10 L2 = 12	 Top view - Depicts the envelope of the tool center point sweep in J1 and J3 while J2 remains at a fixed position of 0°.  Side view - Depicts the envelope of the tool center point sweep in J2 and J3 while J1 remains at a fixed position of 0°.

Configuration Parameters

Logix Designer application can be configured for control of robots with varying reach and payload capacities. As a result, it is very important to know the configuration parameter values for your robot including:

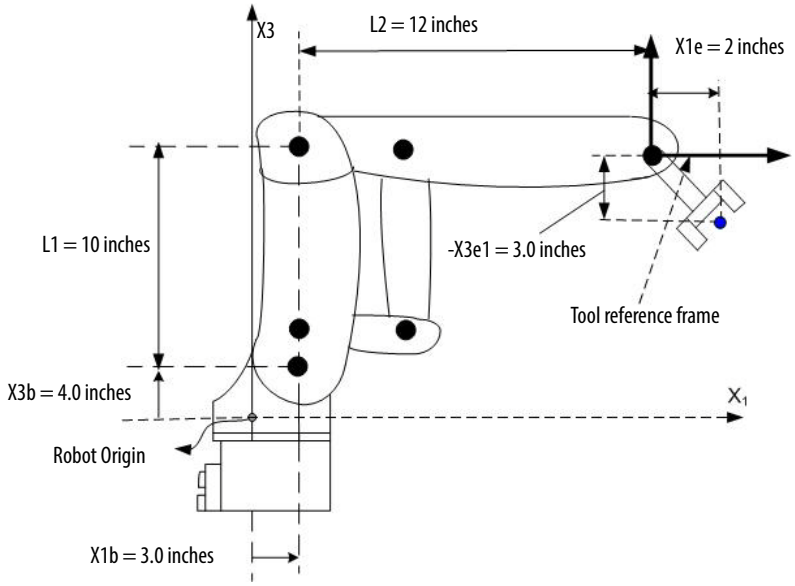
- Link lengths.
- Base offsets.
- End-effector offsets.

The configuration parameter information is available from the robot manufacturer.

IMPORTANT	Verify that the values for the link lengths, base offsets, and end-effector offsets are entered into the Configuration Parameters dialog by using the same measurement units.

This example illustrates the typical configuration parameters for an Articulated Dependent robot.

Figure 44 - Articulated Dependent 4



If the robot is two-dimensional, then X3b and X3e is X2b and X2e respectively.

Link Lengths

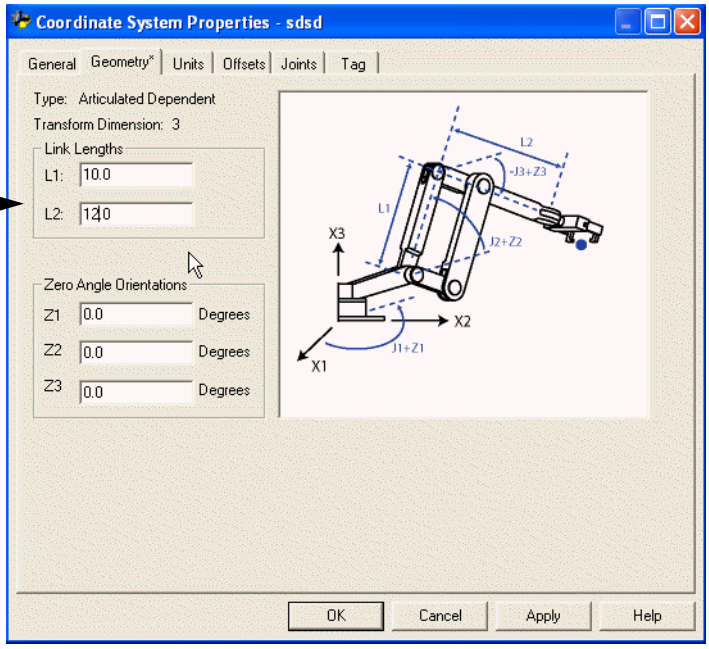
Link lengths are the rigid mechanical bodies attached at joints.

For an articulated dependent robot with	The length of	Is equal to the value of the distance between
Two-dimensions	L1 L2	J1 and J2 J2 and the end-effector
Three-dimensions	L1 L2	J2 and J3 J3 and the end-effector

Figure 45 - Example of Link Lengths for an Articulated Dependent Robot

Enter the Link Length values.
For the robot shown in our example, the Link Length values are:

- L1 = 10.0
- L2 = 12.0



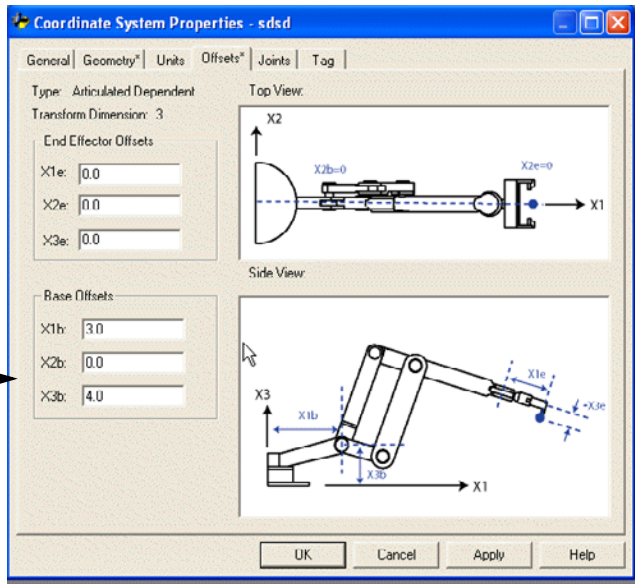
Base Offsets

The base offset is a set of coordinate values that redefines the origin of the robot. The correct base-offset values are typically available from the robot manufacturer. Enter the values for the base offsets in the X1b and X3b fields of the Coordinate System Properties dialog.

Figure 46 - Example of Base Offsets for an Articulated Independent Robot

Enter the Base Offset values.
For the robot shown in our example, the Base Offset values are:

- X1b = 3.0
- X3b = 4.0



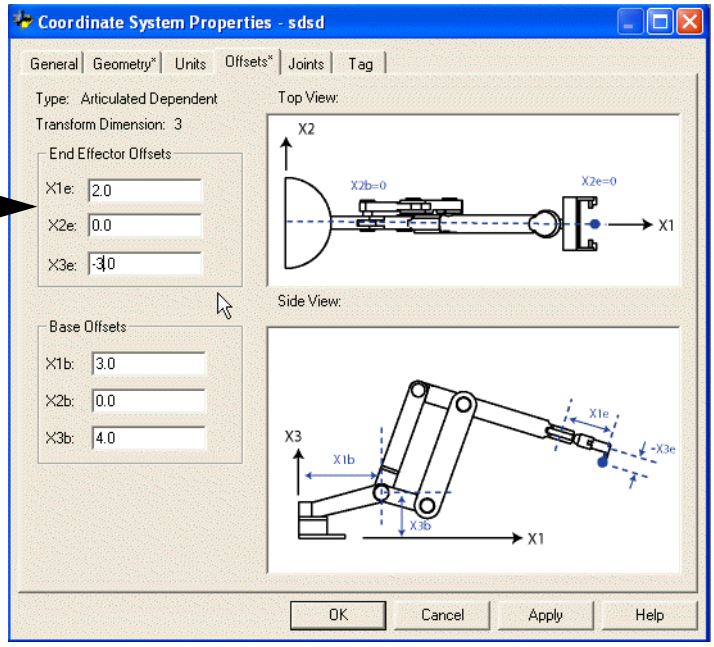
End-effector Offsets

The robot can have an end-effector attached to the end of robot link L2. If there is an attached end-effector, then you must configure the end-effector offset value on the Coordinate System Properties dialog. The end-effector offsets are defined with respect to the tool reference frame at the tool tip.

Figure 47 - Example of End-effector Values for an Articulated Independent Robot

Enter the end-effector offset values.
For the robot shown in our example, the end-effector values are:

- X1e = 2.0
- X3e = -3.0



Notes:

Configure a Cartesian Gantry Robot

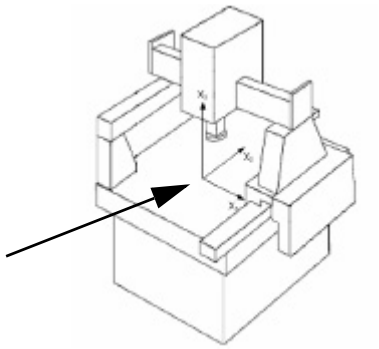
Topic	Page
Establish the Reference Frame for a Cartesian Gantry Robot	103
Identify the Work Envelope for a Cartesian Gantry Robot	104
Define Configuration Parameters for a Cartesian Gantry Robot	104

Use these guidelines when configuring a Cartesian Gantry robot.

Establish the Reference Frame for a Cartesian Gantry Robot

For a Cartesian Gantry robot, the reference frame is an orthogonal set of X1, X2, and X3 axes positioned anywhere on the Cartesian robot. All global coordinate measurements (points) are relative to this reference frame. Typically, the reference frame is aligned with the X1, X2, and X3 axes of the machine

Figure 48 - Cartesian Reference Frame.



To establish a Local coordinate system with axes positions different from the reference frame, use the Motion Redefine Position (MRP) instruction to reset the position register. You can also use the Offset Vector in the MCT transform instruction to establish an offset between the Local coordinate system and the reference frame.

For more information about Motion Instructions, see Logix5000 Controllers Motion Instructions Reference Manual, publication MOTION-RM002.

Identify the Work Envelope for a Cartesian Gantry Robot

The work envelope for a Cartesian Gantry robot is typically a solid rectangle of length, width, and height that is equal to the axis travel limits.

Define Configuration Parameters for a Cartesian Gantry Robot

You do not need to define the link lengths, base offset, or end-effector offset configuration parameters for a Cartesian Gantry robot.

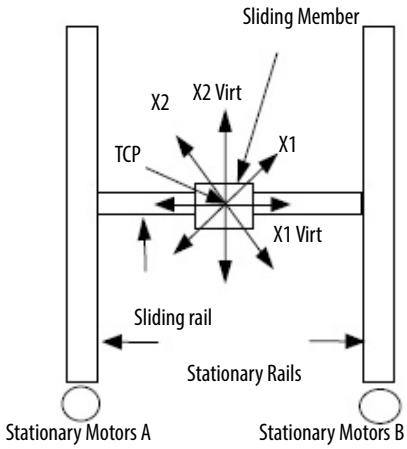
Configure a Cartesian H-bot

Topic	Page
About Cartesian H-bots	105
Establish the Reference Frame for a Cartesian H-bot	106
Identify the Work Envelope for a Cartesian H-bot	106
Define Configuration Parameters for a Cartesian H-bot	107

About Cartesian H-bots

The H-bot is a special type of Cartesian two-axis gantry robot. This type of machine has three rails positioned in the form of a letter H. Two motors are positioned at the end of each leg of the robot. Unlike a standard gantry robot, neither motor is riding on top of the moving rails. Use these guidelines when configuring a Cartesian H-bot.

Figure 49 - Cartesian H-bot



In the Cartesian H-bot illustration above, the X1 and X2 axes are the real axes on the robot. X1 Virt and X2 Virt are configured as the virtual axes.

The configuration of the H-bot mechanical linkages enables it to move at a 45° angle to the axes when either motor A or motor B is rotated.

For example, when:

- Motor A (X1 axis) is rotated, the robot moves along a straight line at + 45° angle
- Motor B (X2 axis) is rotated, the machine moves at an angle of -45°.
- Motors A and B are both rotated clockwise at the same speed, then the machine moves along a horizontal line

- Motors A and B are both rotated counterclockwise at the same speed then the machine moves along a vertical line

Any X,Y position can be reached by properly programming the two motors.

For example, a move of (X1 = 10, X2 = 0) causes the X1X2 axes to move to a position of (X1=7.0711, X2=7.0711). A move to (X1=10, X2 =10) causes the robot to move to a position of (X1=0, X2=14.142).

While this configuration can be very confusing for a programmer, utilizing the Logix Designer application Kinematics function configured with two Cartesian coordinate systems and a -45° rotation easily performs the function.

To configure two Cartesian coordinate systems, Coordinate system 1 (CS1) and Coordinate system 2 (CS2), each containing two linear axes, use the following steps.

1. Configure CS1 to contain the virtual X1 and X2 axes.
2. Configure CS2 to contain the real X1 and X2 axes.
3. Configure the Orientation vector of the MCT instruction as (0,0, -45), a negative degree rotation around the X3 axis.
4. Configure the Translation vector as (0, 0, 0).
5. Link the CS1 and CS2 by using a MCT instruction.
6. Home the H-bot and then program all moves in CS1.

The machine moves the tool center point (TCP) to the programmed coordinates in CS2. The -45° rotation introduced by the Kinematics, counteracts the 45° rotation introduced by the mechanics of the machine and the H-bot moves to the CS1 configured coordinates. As a result, a programmed move of X1virt=10, X2virt=5 moves to a real mechanical position of X1=10, X2=5.

Establish the Reference Frame for a Cartesian H-bot

For a Cartesian H-bot, the Base coordinate system is an orthogonal set of X1, X2 axes postponed anywhere on the Cartesian H-bot. Do not rotate the angular rotation of the reference frame for this robot because the angular rotation vector is used to achieve the 45° rotation required for mechanical operation.

Identify the Work Envelope for a Cartesian H-bot

The work envelope for a Cartesian H-bot is a rectangle of length and width equal to the axis soft travel limits.

Define Configuration Parameters for a Cartesian H-bot

You do not need to define the link lengths, base offset, or end-effector offset configuration parameters for a Cartesian H-bot.

Notes:

Motion Coordinated Instructions

(MCLM, MCCM, MCCD, MCS, MCSD, MCT, MCTP, MCSR, MDCC)

Use this table to choose a motion coordinated instruction.

If you want to	Use this instruction	Available in these languages
Initiate a single or multi-dimensional linear coordinated move for the specified axes within a Cartesian coordinate system.	Motion Coordinated Linear Move (MCLM)	• Relay ladder • Structured text
Initiate a two- or three-dimensional circular coordinated move for the specified axes within a Cartesian coordinate system.	Motion Coordinated Circular Move (MCCM)	• Relay ladder • Structured text
Initiate a change in path dynamics for coordinate motion active on the specified coordinate system.	Motion Coordinated Change Dynamics (MCCD)	• Relay ladder • Structured text
Stop the axes of a coordinate system or cancel a transform.	Motion Coordinated Stop (MCS)	• Relay ladder • Structured text
Initiate a controlled shutdown of all of the axes of the specified coordinate system.	Motion Coordinated Shutdown (MCSD)	• Relay ladder • Structured text
Start a transform that links two coordinate systems together.	Motion Coordinated Transform (MCT)(1)	• Relay ladder • Structured text
Calculate the position of one coordinate system with respect to another coordinate system.	Motion Calculate Transform Position (MCTP)(1)	• Relay ladder • Structured text
Initiate a reset of all of the axes of the specified coordinate system from the shutdown state to the axis ready state and clear the axis faults.	Motion Coordinated Shutdown Reset (MCSR)	• Relay ladder • Structured text
Synchronize one or more motion axes or Coordinate System to a common Master Axis.	Master Driven Coordinate Control (MDCC)	• Relay ladder • Structured text

(1) You cannot use this instruction with SoftLogix(TM) controllers.

Use the motion coordinated instructions to move up to three axes in a coordinate system.

Motion Coordinated Linear Move (MCLM)

Use the MCLM instruction to start a single or multi-dimensional linear coordinated move for the specified axes within a Cartesian coordinate system. You can define the new position as either absolute or incremental.

ATTENTION: Use each tag for the motion control attribute of instructions only once. Re-use of the motion control tag in other instructions can cause unintended operation. This can result in damage to equipment or personal injury.

ATTENTION: Risk of Velocity and/or End Position Overshoot

If you change move parameters dynamically by any method, that is, by changing move dynamics (MCD or MCCD) or by starting a new instruction before the last one has completed, be aware of the risk of velocity and/or end position overshoot.

A Trapezoidal velocity profile can overshoot if maximum deceleration is decreased while the move is decelerating or is close to the deceleration point.

An S-Curve velocity profile can overshoot if either:

- maximum deceleration is decreased while the move is decelerating or close to the deceleration point.
- maximum acceleration jerk is decreased and the axis is accelerating. Keep in mind, however, that jerk can be changed indirectly if it is specified in % of time.

The Motion Coordinated Linear Move (MCLM) instruction performs a linear move by using up to three axes statically coupled as primary axes in a Cartesian coordinate system. You specify whether to use an absolute or incremental target position, the desired speed, maximum acceleration, maximum deceleration, acceleration jerk, deceleration jerk, and the units of each. The actual speed is a function of the programmed units of the speed (Units per sec, or % of Maximum, as configured for the coordinate system), and the combination of primary axes that are commanded to move. Each axis is commanded to move at a speed that lets all axes reach the programmed endpoint (target position) at the same time.

Operands

The MCLM instruction supports the following operands:

- Relay Ladder
- Structured Text

Relay Ladder

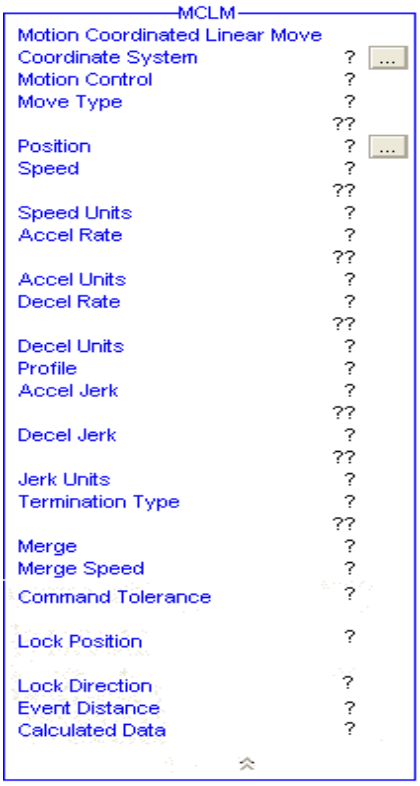


Table 14 - Operands - Relay Ladder

Operand	Type	Format	Description
Coordinate System	COORDINATE_SYSTEM	Tag	The Coordinate System operand specifies the set of motion axes that define the dimensions of a Cartesian coordinate system. The coordinate system supports up to three primary axes. Only those axes configured as primary axes are included in the coordinate velocity calculations.
Motion Control	MOTION_ INSTRUCTION	Tag	Structure used to access instruction status parameters. Refer to Motion Control Bits on page 115.
Move Type	SINT, INT, or DINT	Immediate or tag	Select the Move Type: 0 = Absolute 1 = Incremental Refer to Move Type on page 116.
Position	REAL	Array tag [ ]	A one-dimensional array, whose dimension is defined to be at least the equivalent of the number of axes specified in the coordinate system. The Position array defines either the new absolute or incremental position. [coordination units]
Speed	SINT, INT, DINT, or REAL	Immediate or tag	The Speed operand defines the maximum vector speed along the path of the coordinated move. [coordination units]
Speed Units	SINT, INT, or DINT	Immediate	The Speed Units operand defines the units applied to the Speed operand either directly in coordination units of the specified coordinate system or as a percentage of the maximum values defined in the coordinate system. 0 = Units per Sec 1 = % of Maximum 4 = Units per MasterUnit 7 = Master Units

Table 14 - Operands - Relay Ladder

Operand	Type	Format	Description
Accel Rate	SINT, INT, DINT, or REAL	Immediate or tag	The Accel Rate operand defines the maximum acceleration along the path of the coordinated move. [coordination units]
Accel Units	SINT, INT, or DINT	Immediate	The Accel Units operand defines the units applied to the Accel Rate operand either directly in coordination units of the specified coordinate system or as a percentage of the maximum values defined in the coordinate system. 0 = Units per Sec^2 1 = % of Maximum 4 = Units per $MasterUnit^2$ 7= Master Units
Decel Rate	SINT, INT, DINT, or REAL	Immediate or tag	The Decel Rate operand defines the maximum deceleration along the path of the coordinated move. [coordination units]
Decel Units	SINT, INT, or DINT	Immediate	The Decel Units operand defines the units applied to the Decel Rate operand either directly in coordination units of the specified coordinate system or as a percentage of the maximum values defined in the coordinate system. 0 = Units per Sec^2 1 = % of Maximum 4 = Units per $MasterUnit^2$ 7= Master Units
Profile	SINT, INT, or DINT	Immediate	The Profile operand determines whether the coordinated move uses a trapezoidal or S-Curve velocity profile. 0 = Trapezoidal 1 = S-Curve Refer to Velocity Profiles on page 122.
Accel Jerk	SINT, INT, DINT, or REAL	Immediate or tag	You **must** always enter values for the Accel and Decel Jerk operands. This instruction only uses the values if the Profile operand is configured as S-Curve. Accel Jerk defines the maximum acceleration jerk for the programmed move. For more information on calculating Accel Jerk, see the Jerk Units section below. Decel Jerk defines the maximum deceleration jerk for the programmed move. For more information on calculating Decel Jerk, see the Jerk Units section below. Enter the jerk rates in these Jerk Units. 0 = Units per sec^3 1 = % of Maximum 2 = % of Time 4 = Units per $MasterUnit^3$ 6 = % of Time-Master Driven 7= Master Units Use these values to get started. • Accel Jerk = 100 (% of Time) • Decel Jerk = 100 (% of Time) • Jerk Units = 2 If you want to convert engineering units to % of Time or convert % of Time to engineering units, use the equations shown beginning on page 125.
Decel Jerk	SINT, INT, DINT, or REAL		
Jerk Units	SINT, INT, or DINT		
Termination Type	SINT, INT, or DINT	Immediate or tag	0 = Actual Tolerance 1 = No Settle 2 = Command Tolerance 3 = No Decel 4 = Follow Contour Velocity Constrained 5 = Follow Contour Velocity Unconstrained 6 = Command Tolerance Programmed See Blended Moves and Termination Types on page 34.

Table 14 - Operands - Relay Ladder

Operand	Type	Format	Description
Merge	SINT, INT, or DINT	Immediate	0 = Disabled 1 = Coordinated Motion 2 = All Motion Refer to Merge on page 126.
Merge Speed	SINT, INT, or DINT	Immediate	0 = Programmed 1 = Current
Command Tolerance	REAL	Immediate, real, or tag	The position on a coordinated move is where blending starts. This parameter is used in place of Command Tolerance in the Coordinate System if Termination Type 6 is used. Note that Termination type 2 is identical to Termination Type 6 except the Command Tolerance value from the coordinate system is used and this parameter is ignored.
Lock Position	REAL	Tag	Position on the Master Axis where a Slave starts following the master after the move has been initiated on the Slave Axis.
Lock Direction	UINT32	Immediate, real, or tag	Specifies the conditions for use of the Lock Position.
Event Distance	ARRAY or 0	Array tag	The position(s) on a move measured from the end of the move.
Calculated Data	REAL, ARRAY, or 0	Array tag	Master Distance(s) (or time) needed from the beginning of the move to the Event Distance point.

MCLM(CoordinateSystem, MotionControl, MoveType, Position, Speed, Speedunits, Accelrate, Accelunits, Decelrate, Decelunits, Profile, Acceljerk, Deceljerk, Jerkunits, TerminationType, Merge, Mergespeed, Command Tolerance, Lock Position, Lock Direction, Event Distance, Calculated Data);

Structured Text

The operands for structured text are the same as those for the relay ladder MCLM instruction.

When you enter enumerations for the operand value in structured text, multiple word enumerations must be entered without spaces. For example: enter Decel Units as unitspersec^2 rather than Units per Sec^2 as displayed in the ladder logic.

Use the entries in this table as a guide when entering structured text operands.

Table 15 - Entries for Structured Text Operands

This Operand	Has These Options That You Enter as	
	Text	**Or as**
Coordinate System	No enumeration	Tag
Motion Control	No enumeration	Tag
Move Type	No enumeration	0 (Absolute) 1 (Incremental)
Position	No enumeration	Array tag
Speed	No enumeration	Immediate or tag
Speed Units	Units per sec % of maximum unitspermasterunits masterunits	0 1 4 7
Accel Rate	No enumeration	Immediate or tag
Accel Units	Units per sec^2 % of maximum unitspermasterunits^2 masterunits	0 1 4 7

Table 15 - Entries for Structured Text Operands

This Operand	Has These Options That You Enter as	
	Text	**Or as**
Decel Rate	No enumeration	Immediate or tag
Decel Units	Units per sec^2 % of maximum unitspermasterunits^2 masterunits	0 1 4 7
Profile	Trapezoidal S-Curve	0 1
Accel Jerk	No enumeration	Immediate or tag You **must** always enter a value for the Accel and Decel Jerk operands. This instruction only uses the values if the Profile is configured as S-Curve. Use these values to get started. • Accel Jerk = 100 (% of Time) • Decel Jerk = 100 (% of Time) • Jerk Units = 2
Decel Jerk	No enumeration	
Jerk Units	Unitspersec^3 %ofmaximum %oftime unitspermasternit^3 %oftimemasterdriven masterunits	0 1 2 (use this value to get started) 4 6 7
Termination Type	No enumeration	0 = Actual Tolerance 1 = No Settle 2 = Command Tolerance 3 = No Decel 4 = Follow Contour Velocity Constrained 5 = Follow Contour Velocity Unconstrained 6 = Command Tolerance Programmed See Blended Moves and Termination Types on page 34.
Merge	Disabled Coordinatedmotion Allmotion	0 1 2
Merge Speed	Programmed Current	0 1
Command Tolerance	No enumeration	Immediate or tag
Lock Position	No enumeration	Immediate, real, or tag
Lock Direction	None Immediateforwardonly Immediatereverseonly Positionforward Positionreverse	0 1 2 3 4
Event Distance	No enumeration	Array
Calculated Data	No enumeration	Array

Executing the Instruction

MCLM is a transitional instruction.

- In relay ladder, toggle the rung-condition-in from cleared to set each time the instruction executes.
- In structured text, condition the instruction so that it only executes on a transition.

Motion Control Bits

The following control bits are affected by the MCLM instruction.

Table 16 - Control Bits Affected by the MCLM Instruction

Mnemonic	Description
.EN (Enable) Bit 31	The Enable bit is set when the rung transitions from false to true and resets when the rung goes from true to false.
.DN (Done) Bit 29	The Done bit sets when the coordinated instruction has been verified and queued successfully. Because it is set at the time it is queued, it can appear as set when a runtime error is encountered during the verify operation after it comes out of the queue. It resets when the rung transitions from false to true.
.ER (Error) Bit 28	The Error bit is reset when the rung transitions from false to true. It is set when the coordinated move has not successfully initiated. It is also set with the Done Bit when a queued instruction encounters a runtime error.
.IP (In Process) Bit 26	The In Process bit is set when the coordinated move is successfully initiated. It is reset when: • there is no succeeding move and the coordinated move reaches the new position, or • when there is a succeeding move and the coordinated move reaches the specifications of the termination type, or • when the coordinated move is superseded by another MCLM or MCCM instruction with a merge type of Coordinated Move, or • when terminated by an MCS instruction.
.AC (Active) Bit 23	When you have a coordinated move instruction queued, the Active bit lets you know which instruction is controlling the motion. It sets when the coordinated move becomes active. It is reset when the Process Complete bit is set or when the instruction is stopped.
.PC (Process Complete) Bit 27	The Process Complete bit is reset when the rung transitions from false to true. It is set when there is no succeeding move and the coordinated move reaches the new position, or when there is a succeeding move and the coordinated move reaches the specified termination type.
.ACCEL (Acceleration Bit) Bit 01	The Acceleration bit sets while the coordinated move is in the acceleration phase. It resets while the coordinated move is in the constant velocity or deceleration phase, or when coordinated motion concludes.
.DECEL (Deceleration Bit) Bit 02	The Deceleration bit sets while the coordinated move is in the deceleration phase. It resets while the coordinated move is in the constant velocity or acceleration phase, or when coordinated motion concludes.

Move Type

The Move Type operand specifies the method used to indicate the coordinated move path. There are two Move Types.

Table 17 - Move Types

Move Type	Description
Absolute	The axes move via a linear path to the position defined by the position array at the Speed, Accel Rate and Decel Rate as specified by the operands. When the axis is configured for rotary operation, an Absolute Move type behaves in the same manner as for a linear axis. When the axis position exceeds the Unwind parameter, it is unwound. In this way, axis position is never greater than the Unwind value nor less than zero. The sign of the specified position is interpreted by the interpolator and can be either positive or negative. Negative position values instruct the interpolator to move the rotary axis in a negative direction to obtain the desired absolute position. Positive values indicate that positive motion is desired to reach the target position. When the position value is greater than the unwind value, an error is generated. The axis never moves through more than one unwind cycle before stopping at an absolute position.
Incremental	The coordinate system moves the distance as defined by the position array at the specified Speed, by using the Accel and Decel rates determined by the respective operands, via a linear path. The specified distance is interpreted by the interpolator and can be positive or negative. Negative position values instruct the interpolator to move the axis in a negative direction. Positive values indicate positive motion is desired to reach the target position. Motion greater than one unwind cycle is allowed in Incremental mode.

MCLM Absolute and Incremental Move Type Examples

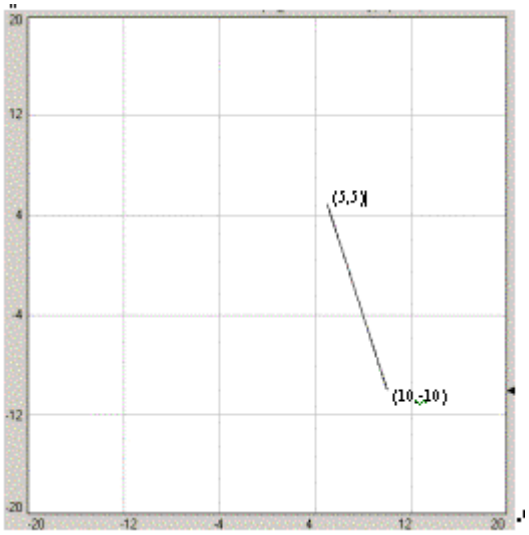
These examples show the use of the MCLM with Move Type of Absolute (first example) and Incremental (second example) to arrive at the same result. The basic assumptions are:

- the two axes, Axis0 and Axis1, are both members of the coordinate system, coordinate_sys.
- Axis0 and Axis1 are orthogonal to each other.
- coordinated_sys is initially at (5,5) units.

Move the Coordinated_sys linearly to (10,-10) units at the vector speed of 10.0 units per second with the acceleration and deceleration values of 5.0 units per second^2.

The following graph is the path generated by the above assumptions.

Figure 50 - Resulting Plot of Path



This is the total distance travelled along the path of the vector.

DAxis0 = 10 - 5 = 5
DAxis1 = -10 - 5 = -15

$$TotalDist = \sqrt{(DAxis0)^2 + (DAxis1)^2} = 15.811388$$

The vector speed of the selected axes is equal to the specified speed in the position units per second. The speed of each axis is proportional to the distance traveled by the axis divided by the square root of the sum of the squares of the distance moved by all axes. The actual speed of Axis0 is the following percent of the vector speed of the move.

%Axis0 Speed = |Daxis0 / TotalDist| = |5 / 15.811388| = .3162 = 31.62%

%Axis1 Speed = |Daxis1 / TotalDist| = |-15 / 15.811388| = .9487 = 94.87%

For the example,

Axis0 Speed = .3162 * 10.0 = 3.162 units/sec.

Axis1 Speed = .9487 * 10.0 = 9.487 units/sec.

The acceleration and deceleration for each axis is the same percentage as speed.

The following ladder instructions show the ladder logic necessary to achieve this path by using Move Type = Absolute and Move Type = Incremental, respectively.

Figure 51 - MCLM Ladder Instruction with Move Type of Absolute

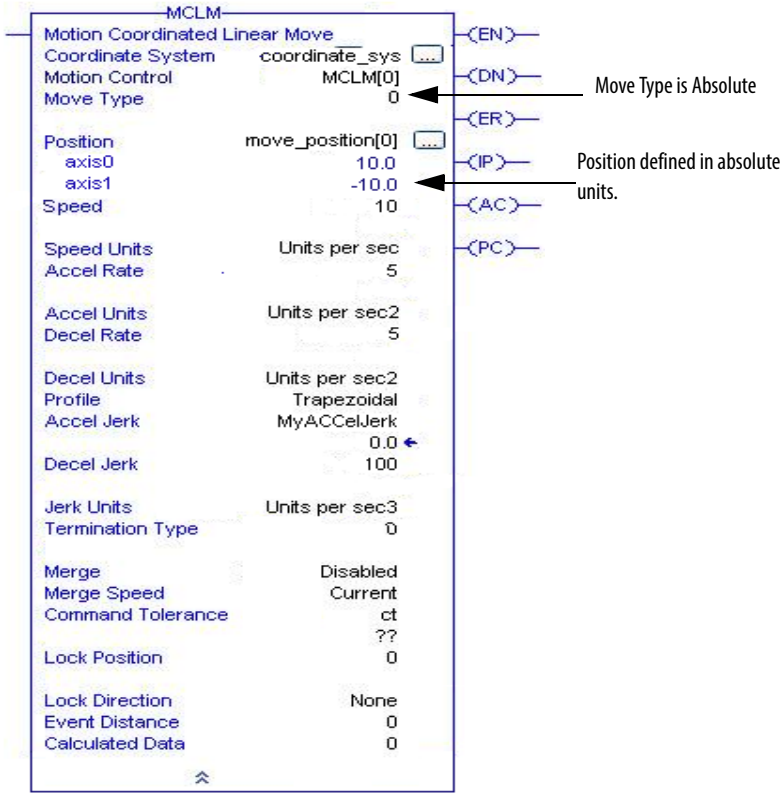
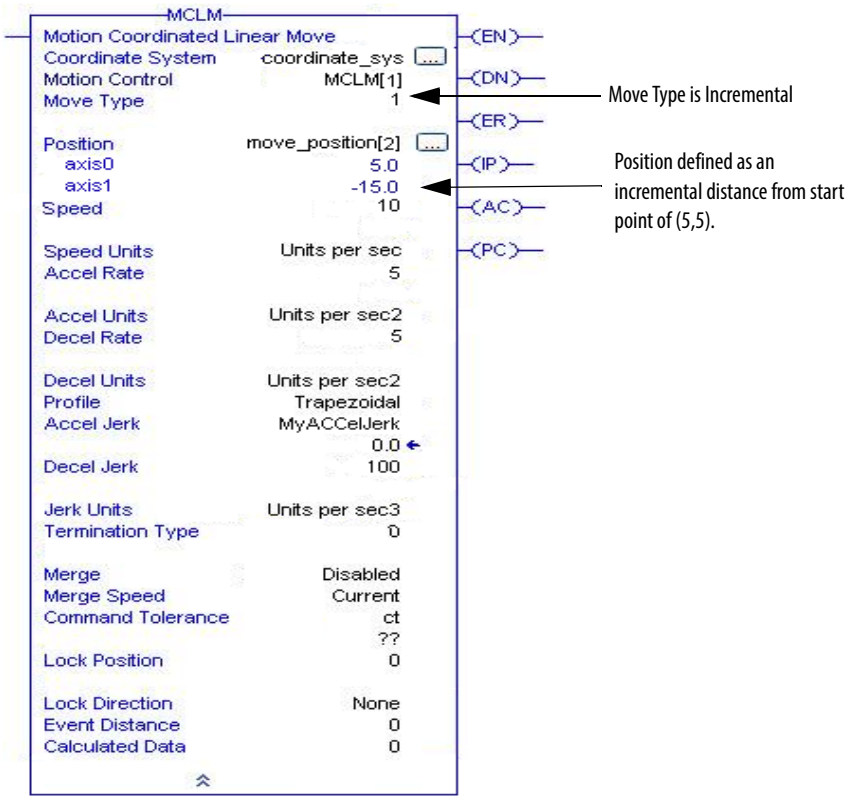


Figure 52 - MCLM Ladder Instruction with Move Type of Incremental

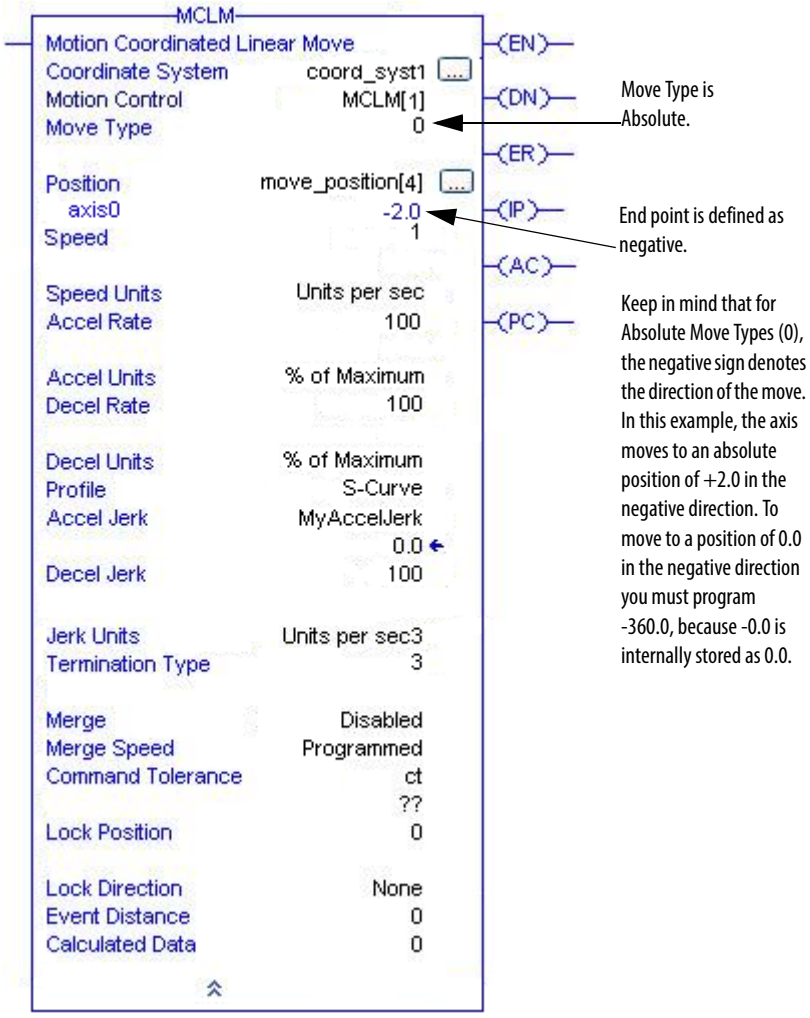


MCLM with One Rotary Axis and Move Type of Absolute

The first example uses a coordinate system of one axis and a Move type of Absolute. The plot of the path is based on the following assumptions:

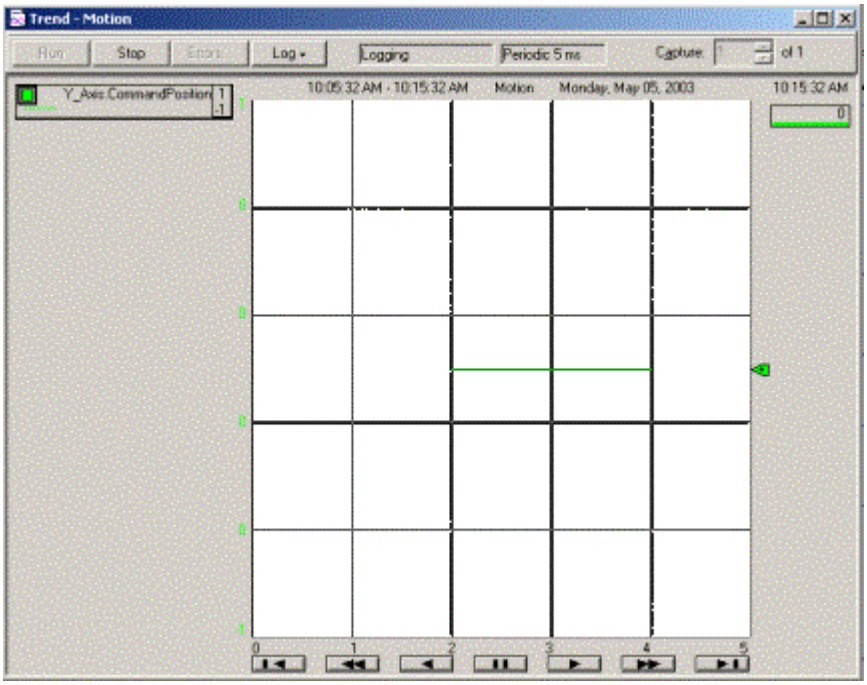
- 1 axis Coordinate System named coord_syst1.
- Axis0 is Rotary with an unwind of 5 revs.
- Start position is 4.
- End position is -2.

Figure 53 - MCLM Ladder Instruction with Move Type of Absolute



The resultant plot of the move's path is shown in the following illustration.

Figure 54 - Plot of MCLM with One Rotary Axis and Move Type of Absolute



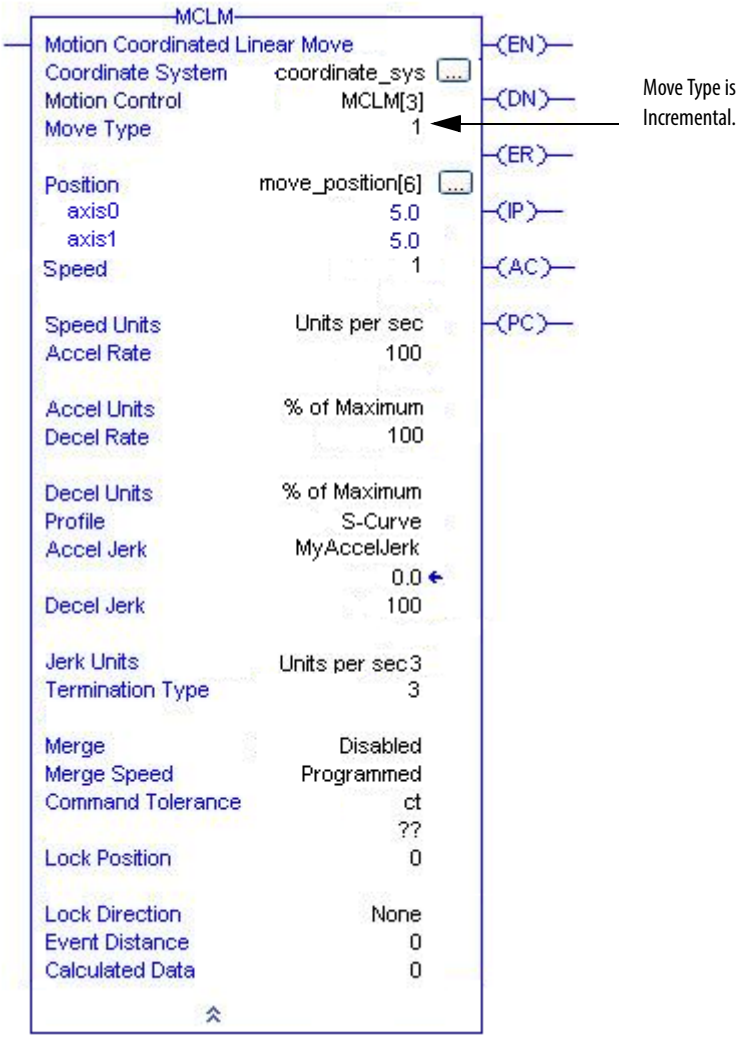
The endpoint was a negative value; therefore, the axis travelled in a negative direction moving from 4 to 2. It did not travel through the unwind. For this move, the endpoint is required to fit within the absolute position defined by the rotary unwind of the axis. Therefore, an unwind value of 6 or -6 is not valid.

MCLM with Two Rotary Axes and Move Type of Incremental

The second MCLM example with rotary axes has two rotary axes and a Move Type of Incremental. The plot of the path has the following assumptions:

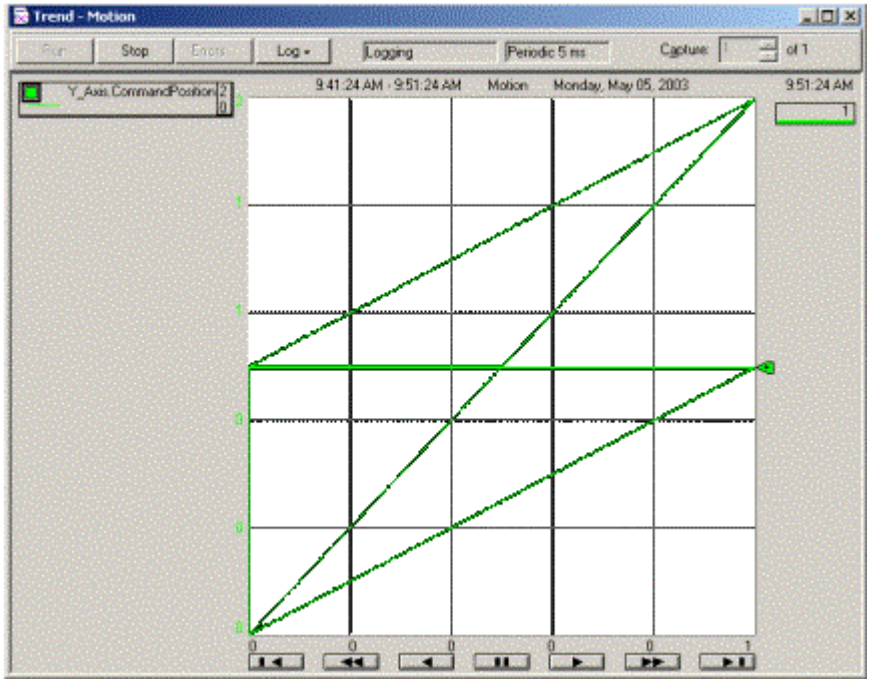
- Two axis Coordinate System named coordinate_sys.
- Axis0 is Rotary with an unwind of 1 revs.
- Axis1 is Rotary with an unwind of 2 revs.
- Start position is 0,0.
- Increment to end position is 5,5.

Figure 55 - MCLM Ladder Instruction with Move Type of Incremental



This MCLM instruction produces the following plot of the moves' path.

Figure 56 - Plot of MCLM with Two Rotary Axes and Move Type of Incremental



In the graphic Plot of MCLM with Two Rotary Axes and Move Type of Incremental, the axes travel a reverse "z" pattern two and one half times, stopping at an actual position of 0,1. This equates to 5 revolutions/unwinds for Axis0 and 2.5 revolutions/unwinds for Axis1. The position increments for this move are positive.

Therefore, the axes move in a positive direction with Axis0 moving from 0 to 1 and Axis1 moving from 0 to 2. In this example, the endpoint is not required to fit within the absolute position defined by the rotary unwind of the axes. The path of the coordinated motion is determined in linear space, but the position of the axes is limited by the rotary configuration.

Velocity Profiles

The Profile operand determines whether the coordinated move uses a trapezoidal or S-Curve velocity profile.

The ControlLogix motion controller provides trapezoidal (linear acceleration and deceleration), and S-Curve (controlled jerk) velocity profiles. A guide to the effects of these motion profiles on various application requirements is given in Table 18.

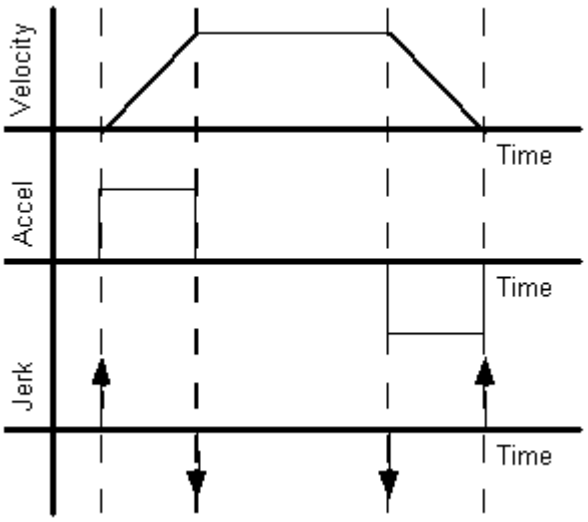
Table 18 - Velocity Profile Effects

Profile Type	**ACC/DEC** Time	**Motor** Stress	**Priority of Control** **Highest to Lowest**			
Trapezoidal	Fastest	Worst	Acc/Dec	Velocity	Position	
S-Curve	2X Slower	Best	Jerk	Acc/Dec	Velocity	Position

Trapezoidal

The trapezoidal velocity profile is the most commonly used profile because it provides the most flexibility in programming subsequent motion and the fastest acceleration and deceleration times. The maximum change in velocity is specified by acceleration and deceleration. Because jerk is not a factor for trapezoidal profiles, it's considered infinite and is shown as series of vertical lines in the following graph.

Figure 57 - Trapezoidal Accel/Decel Time



S-Curve

S-Curve velocity profiles are most often used when the stress on the mechanical system and load needs to be minimized. The S-Curve profile, however, sacrifices acceleration and deceleration time compared to the trapezoidal. The maximum rate at which velocity can accelerate or decelerate is further limited by jerk.

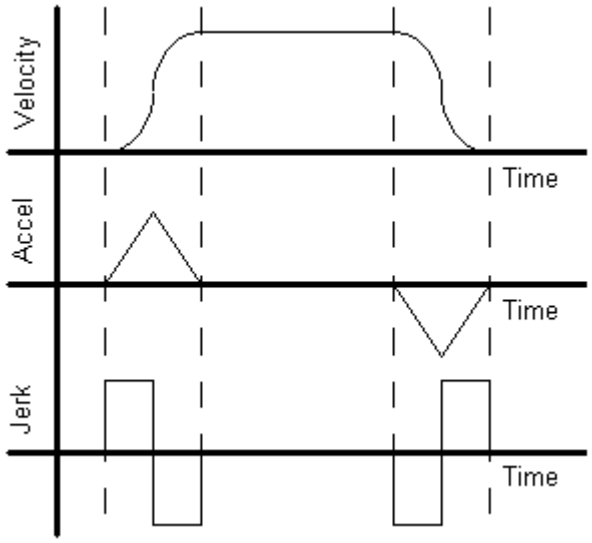
Coordinate motion acceleration and deceleration jerk rate calculations are performed when these instructions are started.

- MAJ

- MAM
- MAS
- MCD
- MCS
- MCCM
- MCLM

The calculated Jerk Rate produces triangular acceleration and deceleration profiles, as shown in the following diagram.

Figure 58 - S-Curve Accel/Decel Time



For an S-Curve move, the Jerk rate is determined based on the programmed velocity, acceleration, and deceleration values, not on the length of the move. Logix Designer application attempts to keep the Jerk rate constant when blending moves that have the same acceleration and deceleration values, even if the move is not long enough to reach the programmed velocity (velocity-limited move).

If an S-Curve Move is Configured as	Then Increasing the Velocity
Not velocity-limited	**Decreases** the execution time of the move
Velocity-limited	**Increases** the execution time of the move

For S-Curve moves that are programmed with a zero speed, the Jerk Rate is determined by the rate of speed programmed for the previous instruction with a non-zero speed.

See the MCCD instruction for more details about the impact changes made by an MCCD instruction.

Convert Jerk Units

The jerk units define the units that are applied to the values entered in the Accel Jerk and Decel Jerk operands. The values are entered directly in the position units of the specified coordinate system or as a percentage. When configured by using % of Maximum, the jerk is applied as a percentage of the Maximum Acceleration Jerk and Maximum Deceleration Jerk operands specified in the coordinate system attributes. When configured by using % of Time, the value is a percentage based on the Speed, Accel Rate, and Decel Rate specified in the instruction.

Convert Engineering Units to a Percentage of Time

If you want to convert engineering units to % of Time, use these equations.
For Accel Jerk:

$$j_a\,[\mathrm{EU/s^3}] = \frac{a_{max}{}^2\,[\mathrm{EU/s^2}]}{v_{max}\,[\mathrm{EU/s}]}\left(\frac{200}{j_a\,[\%\ \mathrm{of\ time}]} - 1\right)$$

For Decel Jerk:

$$j_a\,[\mathrm{EU/s^3}] = \frac{a_{max}{}^2\,[\mathrm{EU/s^2}]}{v_{max}\,[\mathrm{EU/s}]}\left(\frac{200}{j_a\,[\%\ \mathrm{of\ time}]} - 1\right)$$

Convert Percentage of Time to Engineering Units

If you want to convert % of Time to engineering units, use these equations.
For Accel Jerk:

$$j_a\,[\%\ \mathrm{of\ time}] = \frac{2}{1 + \dfrac{j_a\,[\mathrm{EU/s^3}]\ v_{max}\,[\mathrm{EU/s}]}{a_{max}^2\,[\mathrm{EU/s^2}]}}\,100$$

For Decel Jerk:

$$j_d\,[\%\ \mathrm{of\ time}] = \frac{2}{1 + \dfrac{j_d\,[\mathrm{EU/s^3}]\ v_{max}\,[\mathrm{EU/s}]}{d_{max}^2\,[\mathrm{EU/s^2}]}}\,100$$

Merge

The Merge operand determines whether or not to turn the motion of all specified axes into a pure coordinated move.

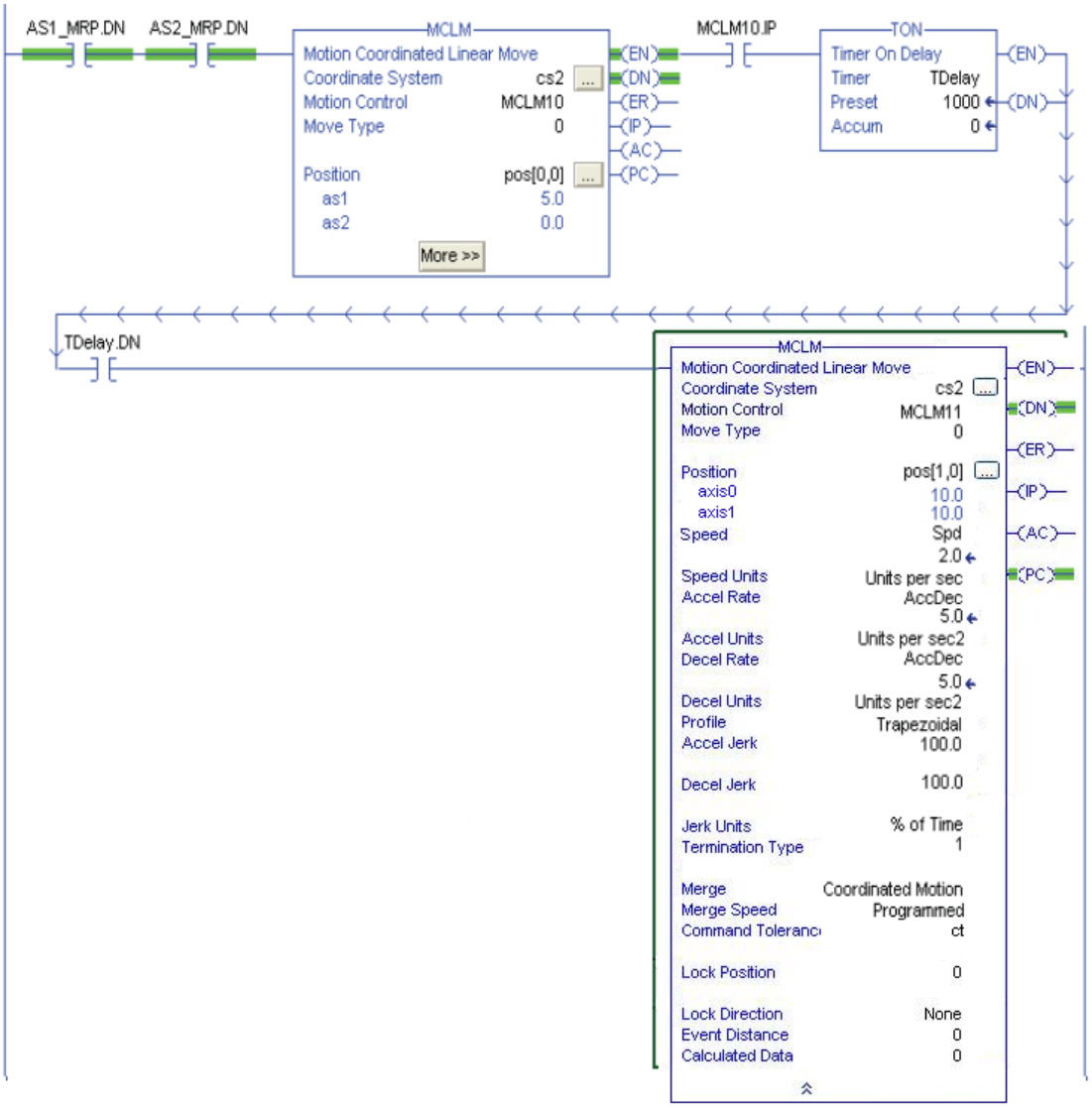
Table 19 - Merge options

Option	Description
Merge Disabled	Any currently executing single axis motion instructions involving any axes defined in the specified coordinate system are not affected by the activation of this instruction, and results in superimposed motion on the affected axes. Also, any coordinated motion instructions involving the same specified coordinate system runs to completion based on its termination type.
Coordinated Motion	Any currently executing coordinated motion instructions involving the same specified coordinate system are terminated. The active motion is blended into the current move at the speed defined in the merge speed parameter. Any pending coordinated motion instructions are cancelled. Any currently executing system single axis motion instructions involving any axes defined in the specified coordinate system is not affected by the activation of this instruction, and results in superimposed motion on the affected axes.
All Motion	Any currently executing single axis motion instructions involving any axes defined in the specified coordinate system and any currently executing coordinated motion instructions are terminated. The prior motion is merged into the current move at the speed defined in Merge Speed parameter. Any pending coordinated move instructions are cancelled.

Coordinated Motion Merge Example

The MCLM ladder diagram uses Coordinate System cs2 to merge an mclm10 instruction with a target absolute position of (5,0) into an mclm11 instruction with the target position of (10,5).

Figure 59 - Ladder Diagram Showing Merge



If the axes are orthogonal to each other, and the coordinate system cs2 is initially at (0,0) units, then the motion caused by this diagram depends on the time at which the second instruction is executed. The blending begins as soon as the second move is initiated and the first move is terminated immediately. In the ladder diagram for this example, transition begins when the timer Tdelay expires.

Figure 60 - Graph Showing Result of Merge

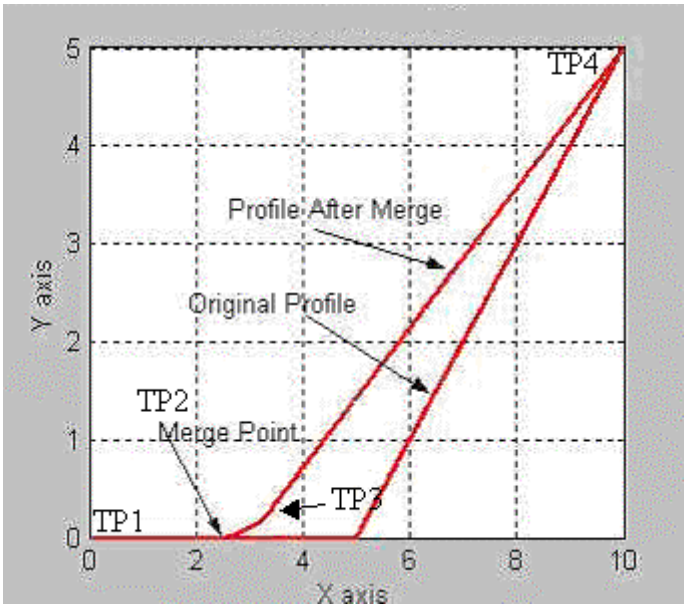


Table 20 - Bit States at Various Transition Points for the Merge Move

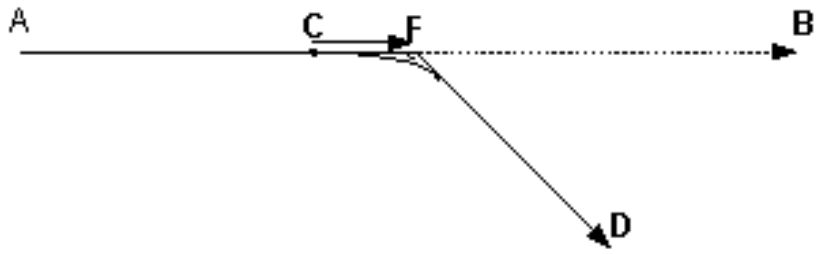
Bit	TP1	TP2	TP3	TP4
Move1.DN	T	T	T	T
Move1.IP	T	F	F	F
Move1.AC	T	F	F	F
mcclm10.PC	F	T	T	T
Move2.DN	T	T	T	T
Move2.IP	T	T	T	F
Move2.AC	F	T	T	F
Move2.PC	F	F	F	T
cs2.MoveTransitionStatus	F	T	F	F
cs2.MovePendingStatus	T	F	F	F
cs2.MovePendingQueueFullStatus	T	F	F	F

Coordinated Motion only supports the queueing of one coordinated motion instruction. Therefore, the MovePendingStatus bit and the MovePendingQueueFullStatus bit are always the same.

Additional Information On Merging Instructions

A move from point A to point B is initiated as shown in the figure below. When the axis is at point C, an incremental merge to point D is initiated. As a result, the current instruction is terminated at point C. The control computes the deceleration distance needed at point C along the vector AB from the current velocity to zero velocity. This distance is shown as vector CF. The imaginary point F is then computed by adding the vector CF to point C. The resultant merged motion from C to D is shown in the illustration below. The motion follows the curved line starting from C then joins the straight line from F to D. Point D is computed from the original point of the merge (point C) by using the incremental data in the merge instruction. This path is identical as if the original programmed move was made from point A to F then from F to D with a termination type of No Decel.

Figure 61 - Merge Example



This example applies to linear merges.

Attempting to merge a circular move can result in programming errors if the resultant path does not define a circle. The circle center in incremental mode is computed from point C (the point of the merge). However, a circle must exist from point F (the computed end of the deceleration) to the end of the merged move.

Merging in Incremental Mode

The Merge for coordinated motion operates differently from a merge on an MAM. For the MCLM, any uncompleted motion at the point of the merge is discarded. For example, assume that you have a single axis MCLM programmed in incremental mode from a starting absolute position = 0 and with the programmed incremental distance = 4 units. If a merge occurs at an absolute position of 1, and the merge is another incremental move of 4 units, the move completes at a position = 5. If this example occurs on a MAM programmed in incremental mode, the final position = 8.

Programming Guidelines for Zero Length Moves

In Master Driven Mode and Time Driven Mode, you have the option of configuring a move with a Slave distance increment of zero (or a move with the same target and current position). If speed is specified in Master Units, the move remains active until the specified Master distance has been traversed. Use this type of move to generate a dwell in a multi-segment path move.

Similarly, when you program the move time directly in seconds in Time Driven Mode, a move of the duration of X seconds with a zero departure results in a programmed delay of the specified time.

IMPORTANT	Instructions with zero length cause velocity of the multi-axis instruction preceding the one with zero length to decelerate to zero at its endpoint. To avoid this behavior, it is suggested that you eliminated moves with zero length from your program.

A zero length move with a duration of zero time completes in the minimum time possible, which is 1 coarse iteration.

Dwells

You have the option to program a dwell by using Time Based Programming in either Time Driven Mode or MDSC Mode when a zero length move (see Zero Length Move below) is programmed. The acceleration, deceleration, and jerk parameters are ignored when a zero length move is programmed. Therefore, when in time driven mode, the duration of the dwell is in seconds. When in MDSC mode, the duration of the dwell is programmed in units of Master Distance.

In MDSC mode, the dwell starts either at the Master Lock Position or immediately, depending on the programmed Lock Direction parameter, and continues for a duration as specified in the Speed parameter.

Time Based Programming Errors

- A zero length move with a duration of zero time completes in 1 coarse iteration, which is the minimum time possible.
- A zero length move that is programmed with Speed Units other than seconds or master distance completes almost immediately.
- An error occurs if a move is programmed by using Time Based Planning that is started with a nonzero velocity. This means that a move using the merge enabled parameter in an instruction causes an error for most cases because merge is typically used when the axes are moving.

- An error occurs if speed is programmed in units of seconds and acceleration, deceleration, or jerk is not programmed in seconds (or % of Time for jerk).

MCLM Target Position Entry Dialog

The Target Position Entry Dialog for the MCLM instruction provides an easy format for editing Position. To gain access to the Target Position Entry dialog box:

- You must have inserted the name of the coordinated system into the instruction,
- You must have a valid tag name entered in the position field with sufficient elements to handle the number of axes, and
- You must have selected a valid Move Type.

To access the MCLM Instruction Target Position Entry Dialog box, press the ellipsis after the Position line on the instruction faceplate.

Figure 62 - MCLM Ladder Valid Values for Accessing Target Position Entry Box

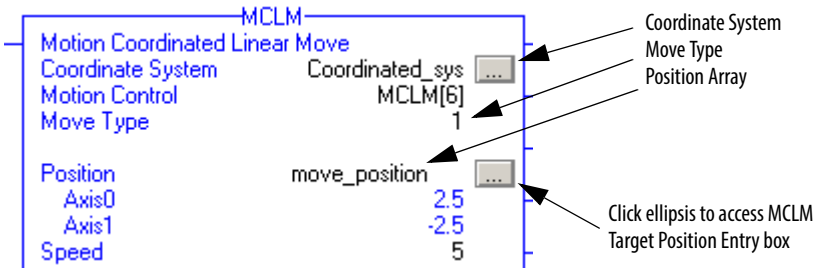
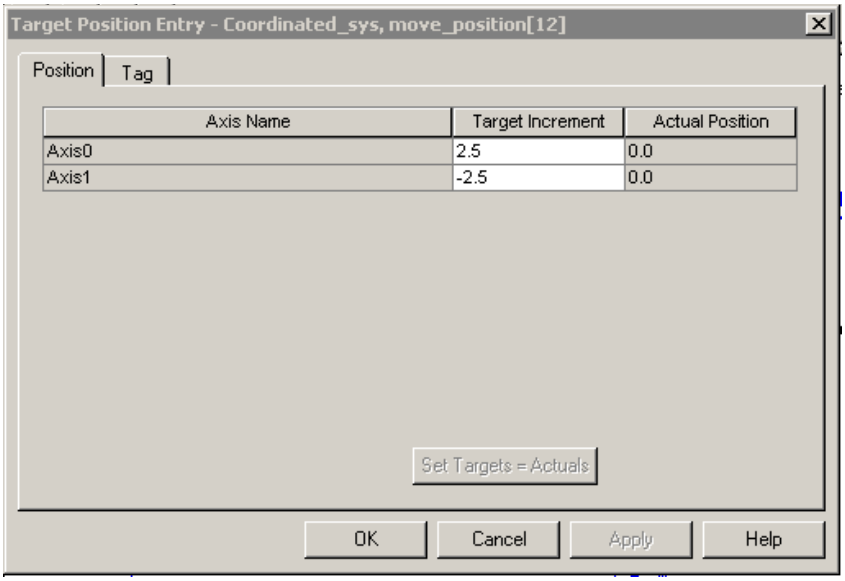


Figure 63 - MCLM Instruction Target Position Entry Dialog - Position Tab



The dialog title indicates the Coordinate System and Tag Names for the instruction.

Table 21 - Target Position Entry Dialog Field Description

Feature	Description
Axis Name	These fields list the names of each axis contained in the Coordinate System. You cannot alter the axis names in this dialog.
Target Position/Target Increment	This field contains the endpoint, or increment, of the coordinated move as specified in the instruction faceplate. It is numeric.
Actual Position	These are the current actual positions of the axes in the coordinate system. These positions are updated dynamically when on-line and Coordinate System Auto Tag Update is enabled.
Set Targets = Actuals Button	This button automatically copies the actual position values to the Target Position Column.

The selected Move type governs the appearance and the availability of the Set Targets = Actuals button.

When the Move Type is Absolute, the target column is entitled Target Position. When the Move Type is Incremental, the target column is entitled Target Increment and the Set Targets = Actuals button is unavailable (grayed out).

Arithmetic Status Flags

Not affected.

Fault Conditions

None.

Error Codes

See Error Codes (ERR) for Coordinate Motion Instructions on page 261.

Runtime Error Conditions

The slave move must start at rest if Speed Units = Seconds or Master Units. Any of the following conditions can cause this error:

- MCLM with Merge = Coordinated Motion or Merge = All Motion and Speed = Seconds or Master Units is started while another MCLM is in progress.
- MCLM uses Term Type = 4 or 5 and Speed = Seconds or Master Units.

Extended Error Codes

Extended Error codes help to further define the error message given for this particular instruction. Their behavior is dependent upon the Error Code with which they are associated.

The Extended Error Codes for Servo Off State (5), Shutdown State (7), Axis Type Not Servo (8), Axis Not Configured (11), Homing In Process Error (16), and Illegal Axis Data type (38) errors all function in the same fashion. A number between 0...*n* is displayed for the Extended Error Code. This number is the index to the Coordinate System indicating the axis that is in the error condition.

For Error Code Axis Not Configured (11) there is an additional value of -1 that indicates the Coordinate System was unable to set up the axis for coordinate motion.

For the MCLM instruction, Error Code 13 - Parameter Out of Range, Extended Errors returns a number that indicates the offending parameter as listed on the faceplate in numerical order from top to bottom beginning with zero. For example, 2 indicates the parameter value for Move Type is in error.

Table 22 - Error Code Indicators and Descriptions

Referenced Error Code and Number	Extended Error Numeric Indicator	Instruction Parameter	Description
Parameter Out Of Range (13)	2	Move Type	Move Type is either less than 0 or greater than 1.
Parameter Out Of Range (13)	3	Position	The position array is not large enough to provide positions for all the axes in the coordinate system.
Parameter Out Of Range (13)	4	Speed	Speed is less than 0.
Parameter Out Of Range (13)	6	Accel Rate	Accel Rate is less than or equal to 0.
Parameter Out Of Range (13)	8	Decel Rate	Decel Rate is less than or equal to 0.
Parameter Out Of Range (13)	11	Termination Type	Termination Type is less than 0 or greater than 3.

Error Code 54 – Maximum Deceleration Value is Zero

If the Extended Error returns a positive number (0-*n*), it's referring to the offending axis in the coordinate system.

1. Go to the Coordinate System Properties General Tab and look under the Brackets ([])column of the Axis Grid to determine which axis has a Maximum Deceleration value of 0.
2. Click the ellipsis next to the offending axis to access the Axis Properties screen.
3. Go to the Dynamics tab and make the appropriate change to the Maximum Deceleration Value.

   If the Extended Error number is -1, this means the Coordinate System has a Maximum Deceleration Value of 0.
4. Go to the Coordinate System Properties Dynamics Tab to correct the Maximum Deceleration value.

MCLM Changes to Status Bits

Status bits provide a means for monitoring the progress of the motion instruction. There are three types of Status bits that provide pertinent information.

- Axis Status bits
- Coordinate System Status bits
- Coordinate Motion Status bits

When the MCLM instruction initiates, the status bits undergo the following changes.

Table 23 - Axis Status Bits

Bit Name	Meaning
CoordinatedMotionStatus	Sets when the instruction starts. Clears when the instruction ends.

Table 24 - Coordinate System Status Bits

Bit Name	Meaning
MotionStatus	Sets when the MCLM instruction is active and the Coordinate System is connected to its associated axes.

Table 25 - Coordinate Motion Status Bits

Bit Name	Meaning
AccelStatus	Sets when vector is accelerating. Clears when a blend is in process or when vector move is decelerating.
DecelStatus	Sets when vector is decelerating. Clears when a blend is in process or when vector move is accelerating.
ActualPosToleranceStatus	Sets for Actual Tolerance termination type only. It sets after the following two conditions are met. 1) Interpolation is complete. 2) The actual distance to the programmed endpoint is less than the configured coordinate system Actual Tolerance value. The bit remains set after an instruction completes. The bit is reset when a new instruction is started.
CommandPosToleranceStatus	Sets for all termination types whenever the distance to the programmed endpoint is less than the configured coordinate system Command Tolerance value. The bit remains set after an instruction completes. It resets when a new instruction is started. The CommandPosToleranceStatus (CS_CMD_POS_TOL_STS) status bit in the Coordinate System is set as follows: TT0, TT1, TT4, TT5 - Bit is set when the distance to the endpoint is less than the Command Tolerance value. The bit is cleared when the first move is complete. TT2, TT6 - Bit is set when the distance to the endpoint is less than the Command Tolerance value. The bit is cleared when the blend is started (that is, when the second move is started). Thus, the bit is not shown if the blend is started at the Command Tolerance (CT) point. The blend can be deferred slightly beyond the CT point if the next move is a short move or for time matching of the acceleration and deceleration of the two adjacent moves. TT3 - Bit is set when the distance to the endpoint is less than the Command Tolerance value (like TT2 and TT6). The bit is cleared when the blend is started. Thus, the bit is not shown if the blend is started at the deceleration pEoint. If the next move is a short move or for time matching of the acceleration and deceleration of the two adjacent moves, it can result in the blend being deferred slightly beyond the deceleration point.
StoppingStatus	The Stopping Status bit is cleared when the MCLM instruction initiates.
MoveStatus	Sets when MCLM begins axis motion. Clears on .PC bit of the last motion instruction or when a motion instruction executes, which causes a stop.
MoveTransitionStatus	Sets when No Decel or Command Tolerance termination type is satisfied. When blending collinear moves, the bit is not set because the machine is always on path. It clears when a blend completes, the motion of a pending instruction starts, or a motion instruction executes, which causes a stop. Indicates not on path.
MovePendingStatus	Sets when one pending coordinated motion instruction is in the instruction queue. Clears when the instruction queue is empty.

Table 25 - Coordinate Motion Status Bits

Bit Name	Meaning
MovePendingQueueFullStatus	Sets when the instruction queue is full. It clears when the queue has room for a new coordinated motion instruction.
CoorMotionLockStatus	Set when an axis lock is requested for an MCLM or MCCM instruction and the axis has crossed the Lock Position. Cleared when an MCLM or MCCM is initiated. For the enumerations Immediate Forward Only and Immediate Reverse Only, the bit is set immediately when the MCLM or MCCM is initiated. When the enumeration is Position Forward Only or Position Reverse Only, the bit is set when the Master Axis crosses the Lock Position in the specified direction. The bit is never set if the enumeration is NONE. The CoordMotionLockStatus bit is cleared when the Master Axis reverses direction and the Slave Axis stops following the Master Axis. The CoordMotionLockStatus bit is set again when the Slave Coordinate System resumes following the Master Axis. The CoordMotionLockStatus bit is also cleared when an MCCS is initiated.

Coordinated Motion only supports the queueing of one coordinated motion instruction. Therefore the MovePendingStatus bit and the MovePendingQueueFullStatus bit are always the same.

Motion Coordinated Circular Move (MCCM)

Use the MCCM instruction to initiate a two or three-dimensional circular coordinated move for the specified axes within a Cartesian coordinate system. New position is defined as either an absolute or incremental position and done at the desired speed. The actual speed of the MCCM is a function of the mode of the move (commanded speed or percent of maximum speed). The speed of the move is based on the time it takes to complete the circular move by using the programmed axes. Each axis is commanded to move at a speed that lets all axes reach the endpoint (target position) at the same time.

The dimension of the circle is defined by the number of axes contained within the coordinate system. For example, if you have a coordinate system that contained three axes with an MCCM instruction that has motion in only two dimensions, the resultant move is still considered a three-dimensional arc or circle.

ATTENTION: Use each tag for the motion control attribute of instructions only once. Re-use of the motion control tag in other instructions can cause unintended operation. This can result in damage to equipment or personal injury.

ATTENTION: Risk of Velocity and/or End Position Overshoot

If you change move parameters dynamically by any method, that is by changing move dynamics (MCD or MCCD) or by starting a new instruction before the last one has completed, be aware of the risk of velocity and/or end position overshoot.

A Trapezoidal velocity profile can overshoot if maximum deceleration is decreased while the move is decelerating or is close to the deceleration point.

An S-Curve velocity profile can overshoot if either:

- maximum deceleration is decreased while the move is decelerating or close to the deceleration point.
- maximum acceleration jerk is decreased and the axis is accelerating. Keep in mind, however, that jerk can be changed indirectly if it is specified in % of time.

Operands

The MCCM instruction supports the following operands:

- Relay Ladder
- Structured Text

Relay Ladder

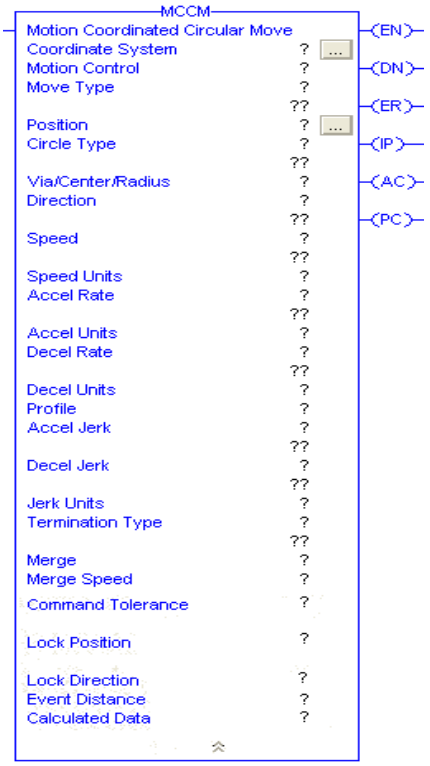


Table 26 - MCCM Instruction Operands - Relay Ladder

Operand	Type	Format	Description
Coordinate System	COORDINATE_SYSTEM	tag	The Coordinate System operand specifies the system of motion axes that define the dimensions of a Cartesian coordinate system. The coordinate system supports up to three primary axes. Only the axes configured as primary axes (up to 3) are included in speed calculations. Only primary axes participate in the actual circular move.
Motion Control	MOTION_ INSTRUCTION	tag	Structure used to access instruction status parameters. Refer to Motion Control Bits on page 144.
Move Type	SINT, INT, or DINT	immediate or tag	0 = Absolute 1 = Incremental Refer to Move Type on page 144.
Position	REAL	array tag[]	A one dimensional array, whose dimension is defined to be at least the equivalent of the number of axes specified in the coordinate system. The Position array defines either the new absolute or incremental position. [coordination units]

Table 26 - MCCM Instruction Operands - Relay Ladder

<table>
<tr><th>Operand</th><th>Type</th><th>Format</th><th>Description</th></tr>
<tr><td>Circle Type</td><td>SINT, INT, or DINT</td><td>immediate or tag</td><td>The Circle Type operand specifies how the array labeled via/center/radius is interpreted. There are four options.
<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td>0=Via</td><td>Indicates that the via/center/radius array members specify a via point between the start and end points.</td></tr>
<tr><td>1=Center</td><td>Indicates that the via/center/radius array members contain the circle center.</td></tr>
<tr><td>2=Radius</td><td>Indicates that the first via/center/radius array member contains the radius. Other members are ignored. Radius is valid only in two-dimensional coordinate systems.</td></tr>
<tr><td>3=Center Incremental</td><td>Indicates that the via/center/radius array members define a position that always incrementally defines the center of the circle regardless of Move Type operand. Sign of the incremental value is measured from the start point to the center.</td></tr>
</table></td></tr>
<tr><td>Via/Center/Radius</td><td>REAL</td><td>array tag[] (via/center)
Immediate or tag (radius)</td><td>Depending on the selected Move Type and Circle Type, the via/center/radius position parameter defines the absolute or incremental value of a position along the circle, the center of the circle or the radius of the circle.
[coordination units]
Refer to Via/Center/Radius on page 145.</td></tr>
<tr><td>Direction</td><td>SINT, INT, or DINT</td><td>immediate or tag</td><td>The Direction operand defines the rotational direction of a 2D circular move as either clockwise or counterclockwise according to the right-hand screw rule. For a 3D circular move, the direction is either Shortest or Longest. In both 2D and 3D, it can also indicate if the circular move is to be a full circle.
<table>
<tr><th>2D</th><th>3D</th></tr>
<tr><td>0=CW</td><td>Shortest</td></tr>
<tr><td>1=CCW</td><td>Longest</td></tr>
<tr><td>2=CW Full</td><td>Shortest Full</td></tr>
<tr><td>3=CCW Full</td><td>Longest Full</td></tr>
</table></td></tr>
<tr><td>Speed</td><td>SINT, INT, DINT, or REAL</td><td>immediate or tag</td><td>The Speed operand defines the maximum vector speed along the path of the coordinated move.
[coordination units]</td></tr>
<tr><td>Speed Units</td><td>SINT, INT, or DINT</td><td>immediate</td><td>The Speed Units operand defines the units applied to the Speed operand either directly in coordination units or as a percentage of the maximum values defined in the coordinate system.
0 = Units per Sec
1 = % of Maximum
3 = Seconds
4= Units per MasterUnit
7 = Master Units</td></tr>
<tr><td>Accel Rate</td><td>SINT, INT, DINT, or REAL</td><td>immediate or tag</td><td>The Accel Rate operand defines the maximum acceleration along the path of the coordinated move.
[coordination units]</td></tr>
<tr><td>Accel Units</td><td>SINT, INT, or DINT</td><td>immediate</td><td>The Accel Units operand defines the units applied to the Accel Rate operand either directly in coordination units of the specified coordinate system or as a percentage of the maximum values defined in the coordinate system.
0 = Units per Sec^2
1 = % of Maximum
3 = Seconds
4= Units per MasterUnit^2
7 = Master Units</td></tr>
</table>

Table 26 - MCCM Instruction Operands - Relay Ladder

Operand	Type	Format	Description
Decel Rate	SINT, INT, DINT, or REAL	immediate or tag	The Decel Rate operand defines the maximum deceleration along the path of the coordinated move. [coordination units]
Decel Units	SINT, INT, or DINT	immediate	The Decel Units operand defines the units applied to the Decel Rate operand either directly in coordination units of the specified coordinate system or as a percentage of the maximum values defined in the coordinate system. 0 = Units per Sec^2 1 = % of Maximum 3 = Seconds 4= Units per $MasterUnit^2$ 7 = Master Units
Profile	SINT, INT, or DINT	immediate	The Profile operand determines whether the coordinated move uses a trapezoidal or an S-Curve velocity profile. Refer to Velocity Profiles on page 122 for more information about Trapezoidal and S-Curve profiles. 0 = Trapezoidal 1 = S-Curve
Accel Jerk	SINT, INT, DINT, or REAL	Immediate or tag	You **must** always enter values for the Accel and Decel Jerk operands. This instruction only uses the values if the Profile is configured as S-Curve. • Accel Jerk defines the maximum acceleration jerk for the programmed move. For more information on calculating Accel Jerk, see Jerk Units section below. • Decel Jerk defines the maximum deceleration jerk for the programmed move. For more information on calculating Decel Jerk, see Jerk Units section below. Enter the jerk rates in these Jerk Units. 0 = Units per sec^3 1 = % of Maximum 2 = % of Time 3 = Seconds 4 = Units per $MasterUnit^3$ 6 = % of Time-Master Driven 7 = Master Units Use these values to get started. • Accel Jerk = 100 (% of Time) • Decel Jerk = 100 (% of Time) • Jerk Units = 2 If you want to convert engineering units to % of Time or convert % of Time to engineering units, use the equations shown beginning on page page 125.
Decel Jerk	SINT, INT, DINT, or REAL	Immediate or tag	
Jerk Units	SINT, INT, or DINT	Immediate or tag	
Termination Type	SINT, INT, or DINT	immediate or tag	0 = Actual Tolerance 1 = No Settle 2 = Command Tolerance 3 = No Decel 4 = Follow Contour Velocity Constrained 5 = Follow Contour Velocity Unconstrained 6 = Command Tolerance Programmed See Blended Moves and Termination Types on page 34.
Merge	SINT, INT, or DINT	immediate	0 = Disabled 1 = Coordinated Motion 2 = All Motion Refer to Merge on page 172.
Merge Speed	SINT, INT, or DINT	immediate	The Merge Speed operand defines whether the current speed or the programmed speed is used as the maximum speed along the path of the coordinated move when Merge is enabled. Current speed is the vector sum of all motion (for example, jogs, MAM's, and geared motion) for all axes defined in the current coordinate system. 0 = Programmed 1 = Current

Table 26 - MCCM Instruction Operands - Relay Ladder

Operand	Type	Format	Description
Command Tolerance	REAL	immediate, real, or tag	The Command Tolerance is the position on a coordinated move where blending starts. This parameter is used in place of Command Tolerance in the Coordinate System if Termination Type 6 is used. Note that termination type 2 is identical to Termination Type 6 except the Command Tolerance value from the coordinate system is used and this parameter is ignored.
Lock Position	REAL	immediate tag	The Lock Position is the position on the Master Axis where a Slave starts following the master after the move has been initiated on the Slave Axis.
Lock Direction	UINT32	immediate, real, or tad	The Lock Direction specifies the conditions in which the Lock Position is used.
Event Distance	ARRAY or 0	array	The Event Distance is the position(s) on a move measured from the end of the move.
Calculated Data	REAL, ARRAY or 0	array	The Calculated Data is the Master Distance(s) (or time) needed from the beginning of the move to the Event Distance point.

MCCM (Coordinate System, Motion Control, Move Type, Position, Circle Type, Via/Center/Radius, Direction, Speed, Speed Units, Accel Rate, Accel Units, Decel Rate, Decel Units, Profile, Accel Jerk, Decel Jerk, Jerk Units, Termination Type, Merge, Merge speed, Command Tolerance, Lock Position, Lock Direction, Event Distance, Calculated Data);

Structured Text

The structured text operands are the same as those for the relay ladder MCCM instruction.

When entering enumerations for the operand value in structured text, multiple word enumerations must be entered without spaces. For example, when entering Decel Units the value is entered as unitspersec^2 rather than Units per Sec^2 as displayed in the ladder logic.

Use the entries in this table as a guide when entering structured text operands.

Table 27 - Entries for Structured Text Operands

This Operand	Has These Options That You Enter as	
	Text	**Or as**
Coordinate System	No enumeration	Tag
Motion Control	No enumeration	Tag
Move Type	No enumeration	Tag 0 = Absolute 1 = Incremental
Position	No enumeration	Array tag
Circle Type	No enumeration	Tag 0 = Via 1 = Center 2 = Radius 3 = Center Incremental
Via/Center/Radius	No enumeration	array tag (via/center) Immediate or tag (radius)

Table 27 - Entries for Structured Text Operands

This Operand	Has These Options That You Enter as	
	Text	**Or as**
Direction	No enumeration	**2D** / **3D** 0 Clockwise / Shortest 1 Counter clockwise / Longest 2 Clockwise full / Shortest full 3 Counter clockwise full / Longest full
Speed	No enumeration	Immediate or tag
Speed Units	Unitspersec %ofmaximum seconds unitspermasterunit masterunits	0 1 3 4 7
Accel Rate	No enumeration	Immediate or tag
Accel Units	Unitspersec^2 %ofmaximum seconds unitspermasterunit^2 masterunits	0 1 3 4 7
Decel Rate	No enumeration	Immediate or tag
Decel Units	Unitspersec^2 %ofmaximum seconds unitspermasterunit^2 masterunits	0 1 3 4 7
Profile	Trapezoidal S-Curve	0 1
Accel Jerk	No enumeration	Immediate or tag You **must** always enter a value for the Accel and Decel Jerk operands. This instruction only uses the values if the Profile is configured as S-Curve. Use these values to get started. • Accel Jerk = 100 (% of Time) • Decel Jerk = 100 (% of Time) • Jerk Units = 2
Decel Jerk	No enumeration	
Jerk Units	Unitspersec^3 %ofmaximum %oftime seconds unitspermasterunit^3 %oftimemasterdriven masterunits	0 1 2 (use this value to get started) 3 4 6 7

Table 27 - Entries for Structured Text Operands

This Operand	Has These Options That You Enter as	
	Text	**Or as**
Termination Type	No enumeration	0 = Actual Tolerance 1 = No Settle 2 = Command Tolerance 3 = No Decel 4 = Follow Contour Velocity Constrained 5 = Follow Contour Velocity Unconstrained 6 = Command Tolerance Programmed See Blended Moves and Termination Types on page 34.
Merge	Disabled Coordinatedmotion Allmotion	0 1 2
Merge Speed	Programmed Current	0 1
Command Tolerance	No enumeration	Immediate or tag
Lock Position	No enumeration	Immediate, real, or tag
Lock Direction	None immediateforwardonly Immediatereverseonly positionforward positionreverse	0 1 2 3 4
Event Distance	No enumeration	Array
Calculated Data	No enumeration	Array

Executing the Instruction

MCCM is a transitional instruction.

- In relay ladder, toggle the rung-condition-in from cleared to set each time the instruction executes.
- In structured text, condition the instruction so that it only executes on a transition.

Motion Control Bits

The following control bits are affected by the MCCM instruction.

Table 28 - Control Bits Affected by MCCM Instruction

Mnemonic	Description
.EN (Enable) Bit 31	The Enable bit is set when the rung transitions from false to true. It resets the rung transitions from true to false.
.DN (Done) Bit 29	The Done bit sets when the coordinated instruction has been verified and queued successfully. Because it is set at the time it is queued, it can appear as set when a runtime error is encountered during the verify operation after it comes out of the queue. It resets when the rung transitions from false to true.
.ER (Error) Bit 28	The Error bit resets when the rung transitions from false to true. It sets when the coordinated move fails to initiate successfully. It can also be set with the Done bit when a queued instruction encounters a runtime error.
.IP (In Process) Bit 26	The In Process bit sets when the coordinated move is successfully initiated. It resets when: • there is a succeeding move and the coordinated move reaches the new position, or • there is no succeeding move and the coordinated move reaches the termination type specifications, or • the coordinated move is superseded by another MCCM or MCLM instruction with a Merge Type of Coordinated Move or • terminated by an MCS or an MCSD instruction.
.AC (Active) Bit 23	When you have a coordinated move instruction queued, the Active bit lets you know which instruction is controlling the motion. It sets when the coordinated move becomes active. It is reset when the Process Complete bit is set or when the instruction is stopped.
.PC (Process Complete) Bit 27	The Process Complete bit resets when the rung transitions from false to true. It sets when: • there is no succeeding move and the coordinated move reaches the new position, or • there is a succeeding move and the coordinated move reaches the termination type specification.
.ACCEL (Acceleration) Bit 01	The Acceleration bit sets while the coordinated move is in acceleration phase. It resets: • while the coordinated move is in the constant velocity or deceleration phase, or • when coordinated motion concludes.
.DECEL (Deceleration) Bit 02	The Deceleration bit sets while the coordinated move is in deceleration phase. It resets: • while the coordinated move is in the constant velocity or acceleration phase, or • when coordinated motion concludes.

Move Type

The Move Type operand determines the method used by the position array to indicate the path of the coordinated move and the method the via/center/radius parameter uses to indicate the via and center circle positions. There are two options.

Figure 64 - Move Type Descriptions

Option	Description
Absolute	The coordinate system moves to the specified Position at the defined Speed, by using the Accel and Decel Rates as designated by their respective operands, along a circular path. When an axis is configured for rotary operation, absolute moves are handled in the same manner as with linear axes. When the axis position exceeds the Unwind parameter, an error is generated. The sign of the specified position array is interpreted by the controller as the direction for the move. Negative position values instruct the interpolator to move the rotary axis in a negative direction to obtain the desired absolute position. A positive value indicates that positive motion is desired to reach the target position. To move to the unwind position in the negative direction, a negative unwind position value must be used as 0 and -0 are treated as 0. When the position is greater than the unwind value, an error is generated. The axis can move through the unwind position but never incrementally more than one unwind value.
Incremental	The coordinate system moves the distance as defined by the position array at the specified Speed, by using the Accel and Decel rates determined by the respective operands, along a circular path. The specified distance is interpreted by the interpolator and can be positive or negative. Negative position values instruct the interpolator to move the rotary axis in a negative direction, while positive values indicate positive motion is desired to reach the target position.

Via/Center/Radius

The via/center/radius position parameter defines the absolute or incremental value of a position along the circle, the center of the circle, or the radius of the circle in relation to the Move and Circle Types, as defined in the following table. If the Circle Type is via or center, the via/center/radius position parameter is a one-dimensional array, whose dimension is defined to be at least the equivalent of the number of axes specified in the coordinate system. If the Circle type is radius, the via/center/radius position parameter is a single value.

Table 29 - Behavior by Type

Move Type	Circle Type	Behavior
Absolute	Via	The via/center/radius position array defines a position along the circle. For a non-full circle case, the Position parameter array defines the endpoint of the arc. For a full circle case, the Position parameter array defines any second point along the circle except the endpoint.
Incremental	Via	The sum of the via/center/radius position array and the old position defines the position along the circle. For a non-full circle case, the sum of the Position parameter array and the old position defines the endpoint of the arc. For a full circle case, the sum of the Position parameter array and the old position defines any second point along the circle except the endpoint.
Absolute	Center	The via/center/radius position array defines the center of the circle. For a non-full circle case, the Position parameter array defines the endpoint of the arc. For a full circle case, the Position parameter array defines any second point along the circle except the endpoint.
Incremental	Center	The sum of the via/center/radius position array and the old position defines the center of the circle. For a non-full circle case, the sum of the Position parameter array and the old position defines the endpoint of the arc. For a full circle case, the sum of the Position parameter array and the old position defines any second point along the circle except the endpoint.

Table 29 - Behavior by Type

Move Type	Circle Type	Behavior
Absolute or Incremental	Radius	The via/center/radius position single value defines the arc radius. The sign of the value is used to determine the center point to distinguish between the two possible arcs. A positive value indicates a center point that generates an arc less than 180 degrees. A negative value indicates a center point that generates an arc greater than 180 degrees. This Circle Type is only valid for two-dimensional circles. The position parameter array follows the Move Type to define the endpoint of the arc.
Absolute	Center Incremental	The sum of the via/center/radius position array and the old position defines the center position of the circle. For a non-full circle case, the Position parameter array defines the endpoint of the arc. For a full circle case, the Position parameter array defines any second point along the circle except the endpoint.
Incremental	Center Incremental	The sum of the via/center/radius position array and the old position defines the center position of the circle. For a non-full circle case, the sum of the Position parameter array and the old position defines the endpoint of the arc. For a full circle case, the sum of the Position parameter array and the old position defines any second point along the circle except the endpoint.

Two-Dimensional Arc and Circle Examples

The following examples show the use of Absolute and Incremental Move Types with the various Circle Types.

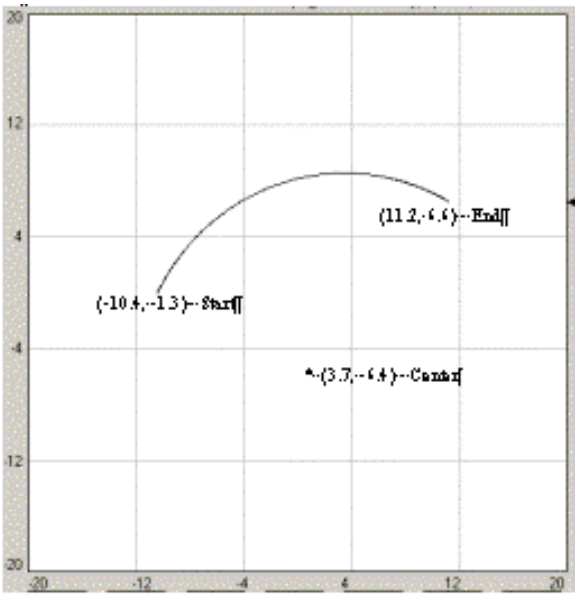
MCCM Using Center Circle Type

The following examples show the use of the MCCM instruction with a Circle Type of Center and a Move Type of Absolute (first example) and Incremental (second example) to arrive at the same result. The basic assumptions are:

- the two axes, Axis0 and Axis1, are both members of the coordinate system, Coordinated_sys.
- Axis0 and Axis1 are orthogonal to each other.
- coordinated_sys is initially at (-10.4,-1.3) units.

Move Coordinated_sys along an arc to (11.2,6.6) units with a center of (3.7,-6.4) units at the vector speed of 10.0 units per second with the acceleration and deceleration values of 5.0 units per second^2. The following graph shows the path generated by the preceding information.

Figure 65 - Plot of MCCM Instruction with Circle Type of Center.



The vector speed of the selected axes is equal to the specified speed in the units per second or percent of the maximum speed of the coordinate system. Likewise, the vector acceleration and deceleration is equal to the specified acceleration/deceleration in the units per second^2 or percent of maximum acceleration of the coordinate system.

This path can be achieved by using an MCCM instruction in the clockwise direction with a Move Type = Absolute or with a Move Type = Incremental. When a Circle Type of Center is chosen, the Via/Center/Radius position defines the center of the arc.

Figure 66 - MCCM Ladder Instruction with Move Type of Absolute

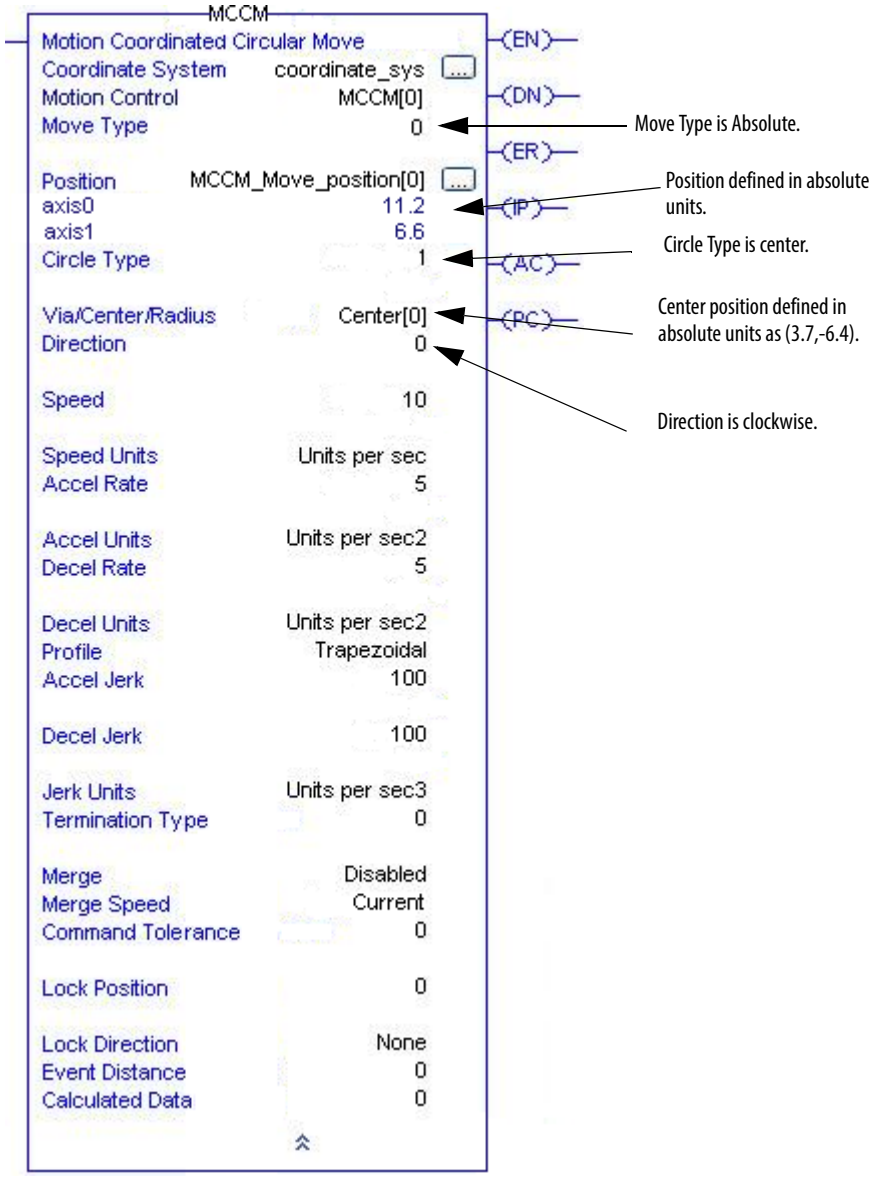
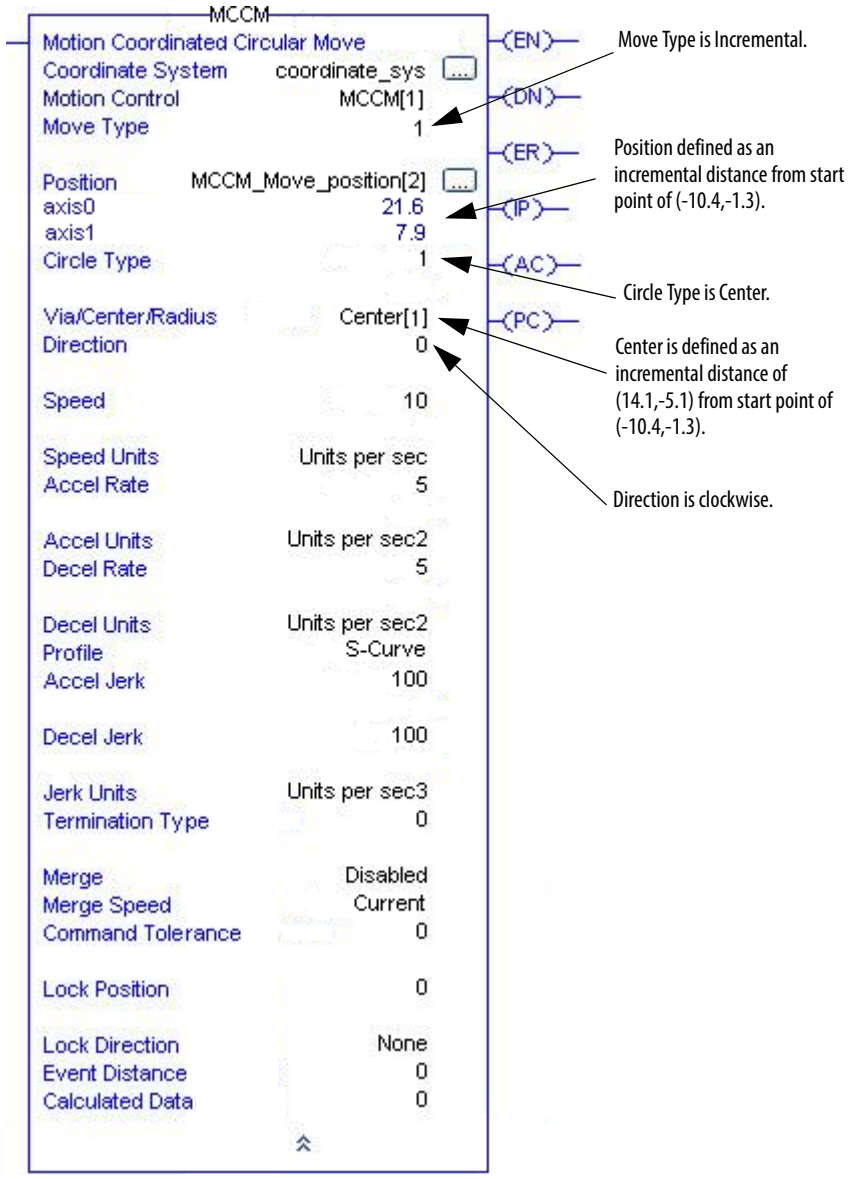
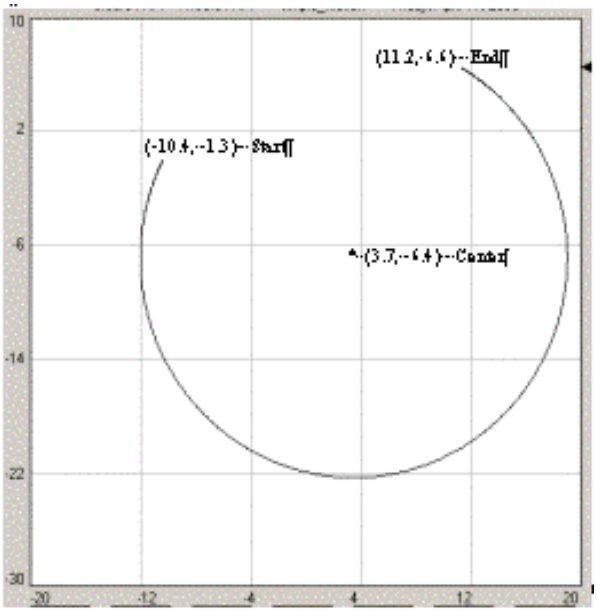


Figure 67 - MCCM Ladder Instruction with Move Type of Incremental



Had a Direction of Counterclockwise been selected (Direction = 1), the axes move along the curve shown in the following graph.

Figure 68 - Plot of Path with Direction as Counterclockwise



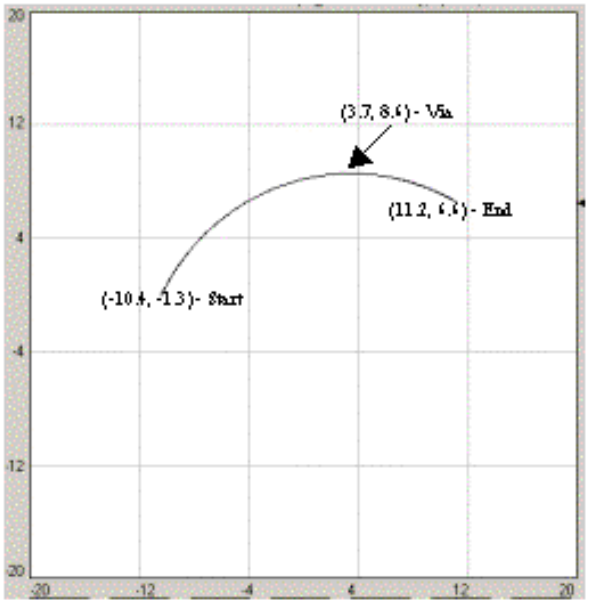
MCCM Instruction Using Via Circle Type

The following examples show the use of the MCCM instruction with a Circle Type of Via and a Move Type of Absolute (first example) and Incremental (second example) to arrive at the same result. The basic assumptions are:

- the two axes, Axis0 and Axis1, are both members of the coordinate system, coordinate_sys.
- Axis0 and Axis1 are orthogonal to each other.
- Coordinated_sys is initially at (-10.4,-1.3) units.

Move Coordinated_sys along an arc to (11.2,6.6) units passing through the point (3.7,8.6) units at the vector speed of 10.0 units per second with the acceleration and deceleration values of 5.0 units per second^2. The following graph shows the path generated by the preceding information.

Figure 69 - Plot of Path of MCCM Instruction with Operands of Via and Absolute



The vector speed of the selected axes is equal to the specified speed in the units per second or percent of the maximum speed of the coordinate system. Likewise, the vector acceleration and deceleration is equal to the specified acceleration/deceleration in the units per second^2 or percent of maximum acceleration of the coordinate system.

This path can be achieved by using an MCCM instruction in the Clockwise direction with a Move Type = Absolute or with a Move Type = Incremental. When a Circle Type of Via is chosen, the Via/Center/Radius position defines a point through which the arc must pass.

Figure 70 - MCCM Ladder Instruction with Operand Values of Via and Absolute

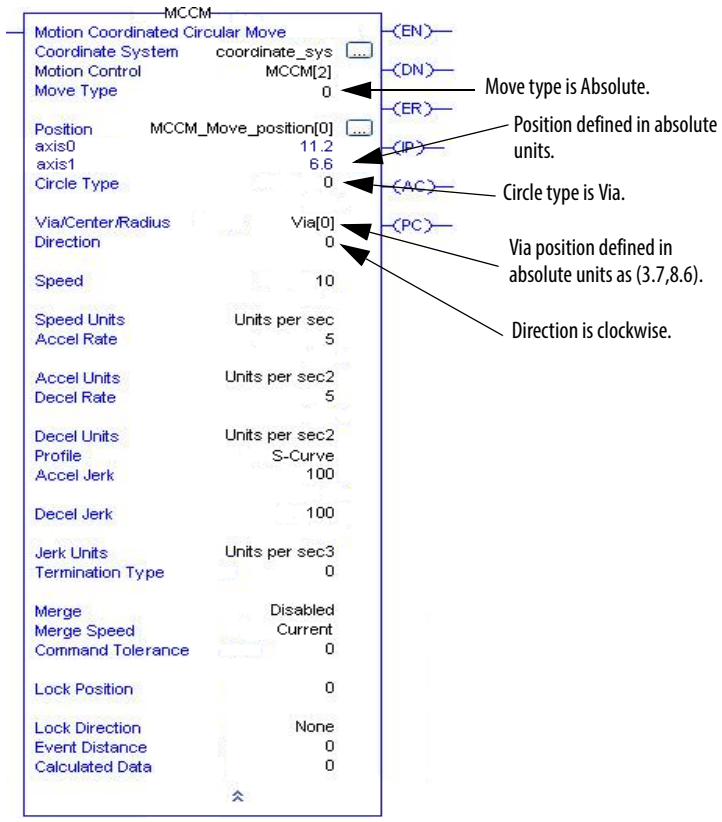
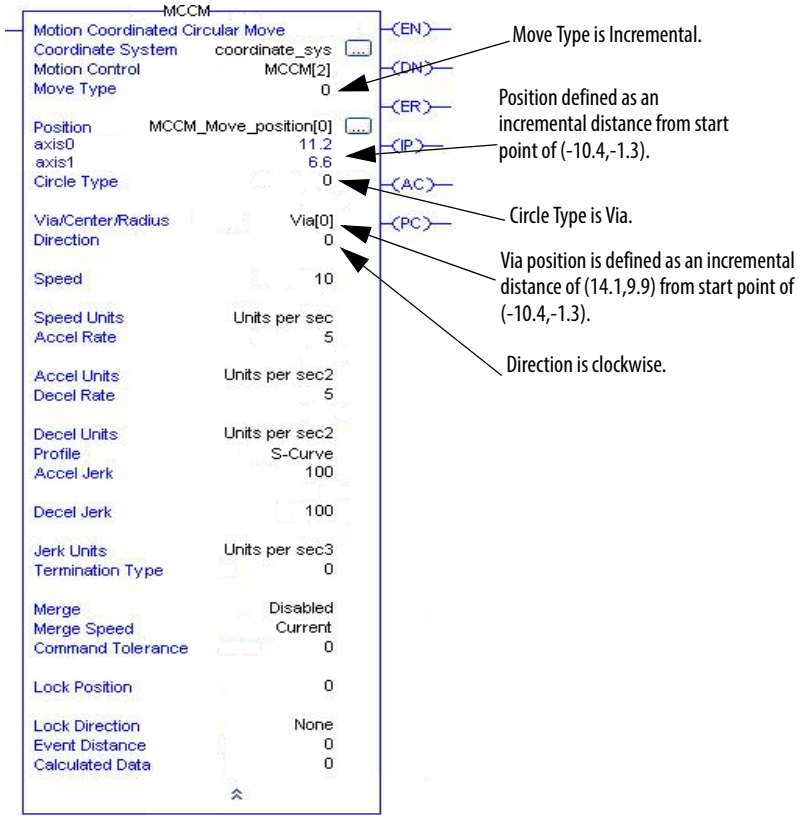


Figure 71 - MCCM Ladder Instruction with Operand Values of Via and Incremental



There are three points (the current position of the axes, the specified end point, and the specified via point), making it difficult to program a bad arc. While it is still certainly possible to program an arc that is not the intended one, a Circular Programming Error runtime fault occurs with an arc if the three points are co-linear (all three on the same line) or not unique (two or more points are the same). In addition, the via point implies that the direction of the arc and thus, it is not necessary (and is ignored) to specify the direction.

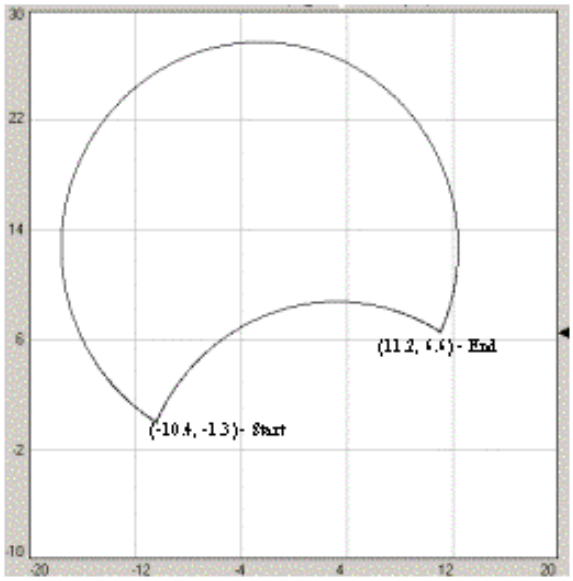
MCCM Instruction Using Radius Circle Type

The following examples show the use of the MCCM instruction with a Circle Type of Radius and a Move Type of Absolute (first example) and Incremental (second example) to arrive at the same result. The basic assumptions are:

- the two axes, Axis0 and Axis1, are both members of the coordinate system, coordinate_sys.
- the coordinate system dimension value is configured as 2. Radius Circle Types can only be configured when two dimensions are configured for the coordinate system.
- Axis0 and Axis1 are orthogonal to each other.
- coordinate_sys is initially at (-10.4,-1.3) units.

Move Coordinated_sys along an arc to (11.2,6.6) units with a radius of 15 units at the vector speed of 10.0 units per second with the acceleration and deceleration values of 5.0 units per second^2. The following graph shows the paths generated by the preceding information by using a Radius value of 15 units and -15 units.

Figure 72 - Plot of Path with Circle Type of Radius



This path can be achieved by using an MCCM instruction in the Clockwise direction with a Move Type = Absolute or with a Move Type = Incremental. When a Circle Type of Radius is chosen, the Via/Center/Radius position defines the radius of the arc.

Figure 73 - MCCM Instruction Move Type Absolute; Circle Type Radius

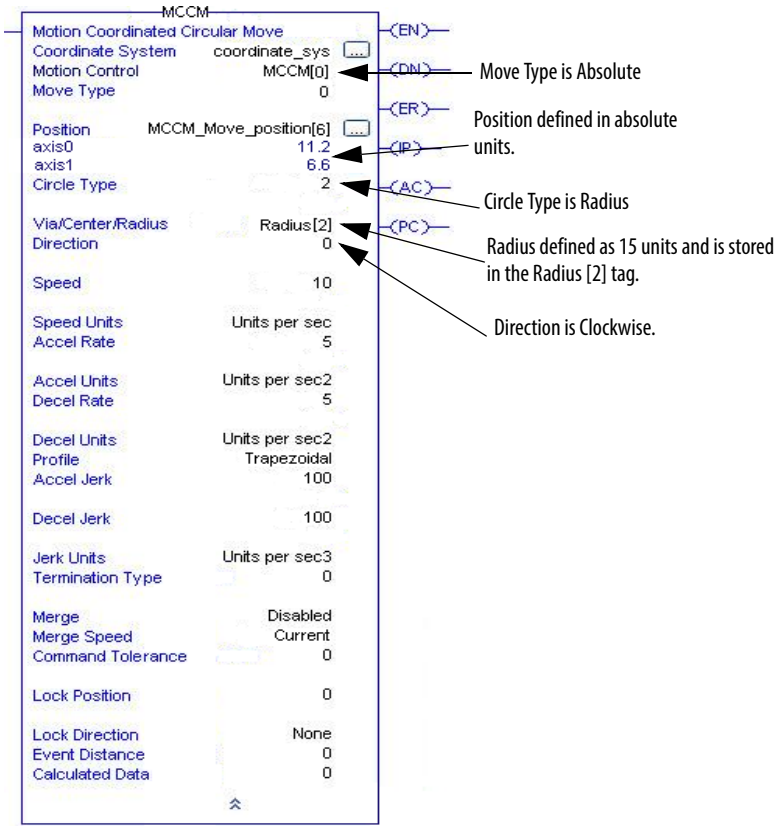
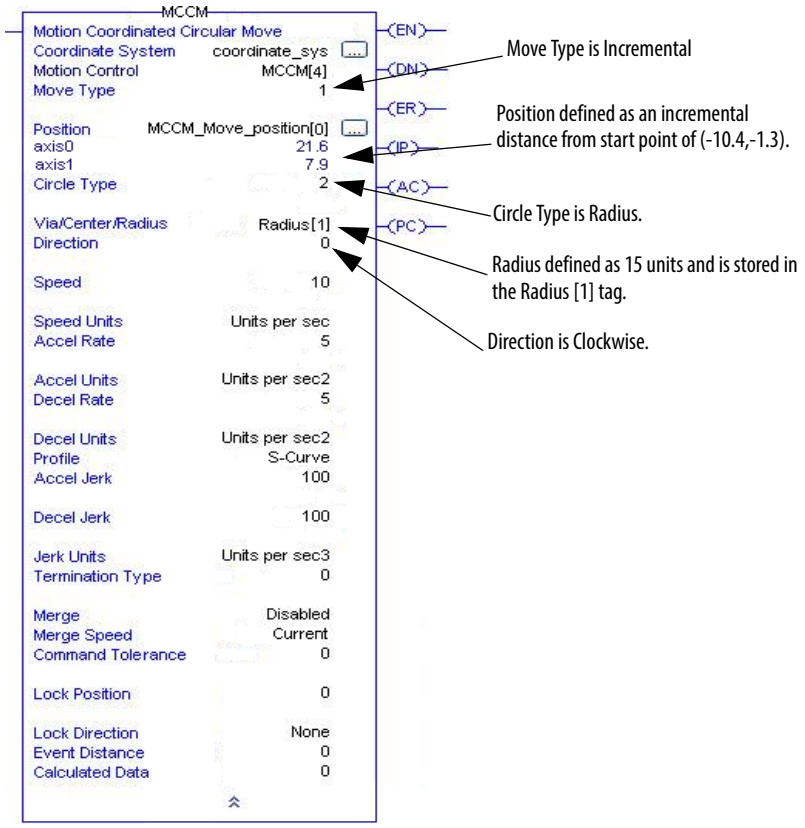


Figure 74 - MCCM Instruction Move Type Incremental; Circle Type Radius



The Move Type has no effect on the Radius value specification. A Positive radius always creates a shorter (<180°) arc and a negative radius creates a longer (>180°) arc (see path graph). If it is 180°, the sign of the radius is irrelevant. A Circle Type of Radius is valid in two-dimensional coordinate systems only.

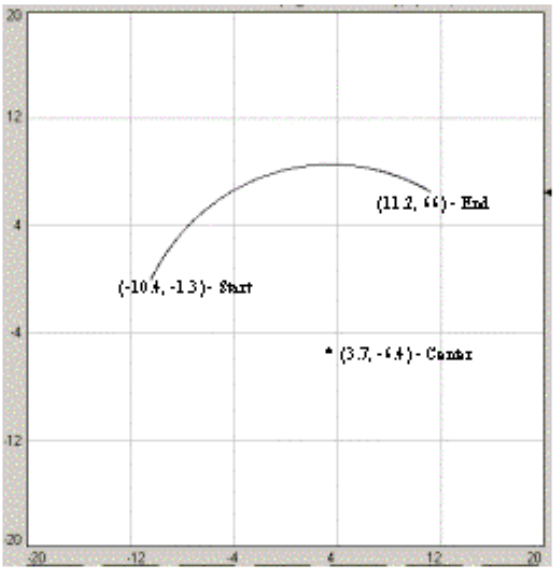
MCCM Using Center Incremental Circle Type

The following examples show the use of the MCCM instruction with a Circle Type of Center Incremental and a Move Type of Absolute (first example) and Incremental (second example) to arrive at the same result. The basic assumptions are:

- the two axes, Axis0 and Axis1, are both members of the coordinate system, coordinate_sys.
- Axis0 and Axis1 are orthogonal to each other.
- coordinate_sys is initially at (-10.4,-1.3) units.

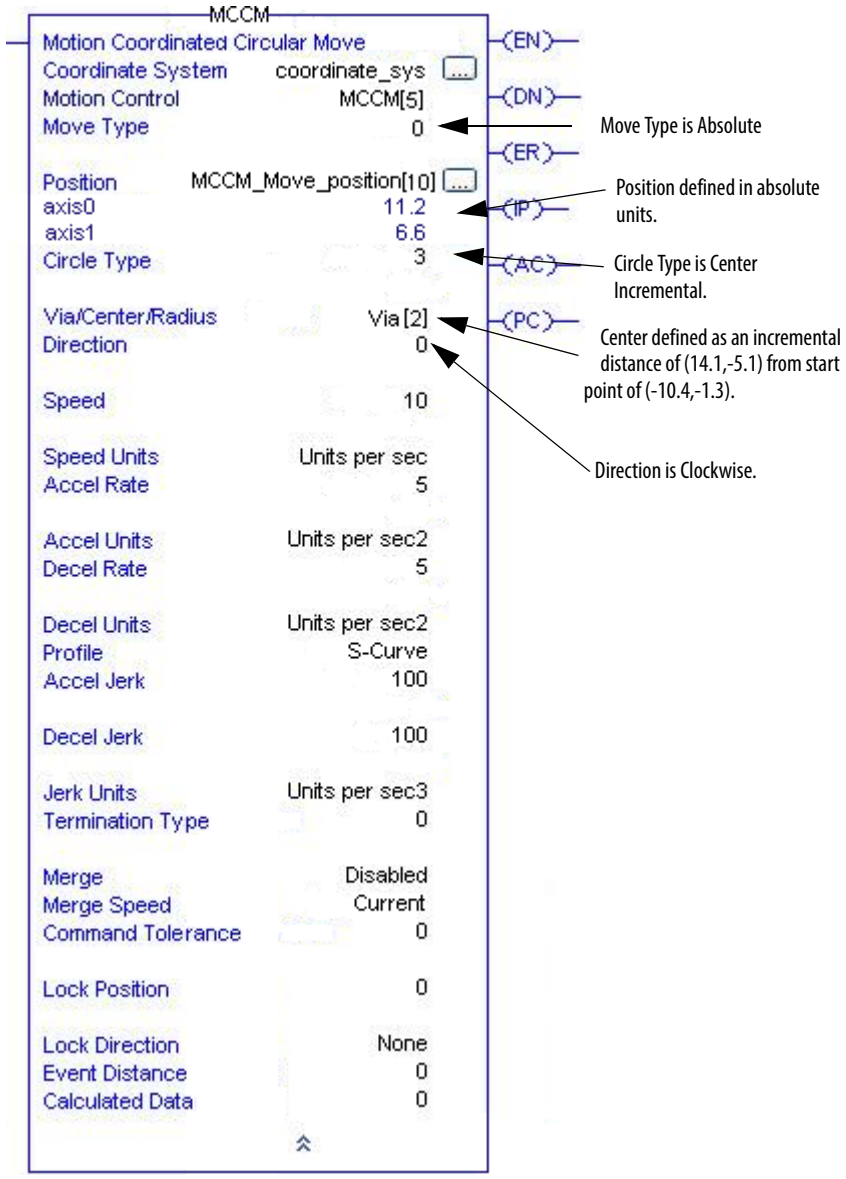
Move coordinate_sys along an arc to (11.2,6.6) units with a center at an increment of (14.1,-5.1) units from the start point at the vector speed of 10.0 units per second with the acceleration and deceleration values of 5.0 units per second^2. The following graph shows the path generated by the preceding information.

Figure 75 - Plot of Path with Circle Type of Center Incremental



This path can be achieved by using an MCCM instruction in the Clockwise direction with a Move Type = Absolute or with a Move Type = Incremental. When a Circle Type of Center Incremental is chosen, the Via/Center/Radius position defines the center of the arc.

Figure 76 - MCCM Instruction Move Type Absolute; Circle Type Center Incremental



The MCCM instruction with Move Type of Incremental and Center Type of Center Incremental is the same as an MCCM instruction with Move Type Incremental and Circle Type of Center.

MCCM Using Two-Dimensional Full Circle

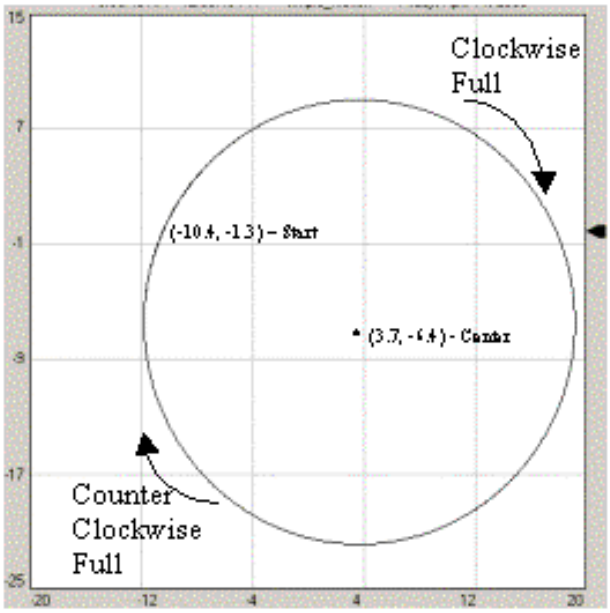
Creating a full circle is a special case of a circular arc. The following is an example of a two-dimensional full circle.

The following examples show the use of the MCCM instruction with a Circle Type of Center and a Move Type of Absolute (first example) and Incremental (second example) to create a full circle. The basic assumptions are:

- The two axes, Axis0 and Axis1, are both members of the coordinate system, Coordinated_sys.
- Axis0 and Axis1 are orthogonal to each other.
- Coordinated_sys is initially at (-10.4,-1.3) units.

Move Coordinated_sys along an arc to (-10.4,-1.3) units with a center at (3.7,-6.4) units from the start point at the vector speed of 10.0 units per second with the acceleration and deceleration values of 5.0 units per second^2. The following graph shows the circle generated by the preceding information.

Figure 77 - Plot of Path of MCCM Instruction Full Circle



This path can be achieved by using an MCCM instruction in the Clockwise direction with a Move Type = Absolute or with a Move Type = Incremental. When a Circle Type of Center is chosen, the Via/Center/Radius position defines the center of the arc.

Figure 78 - MCCM Instruction Move Type Absolute; Circle Type Center.

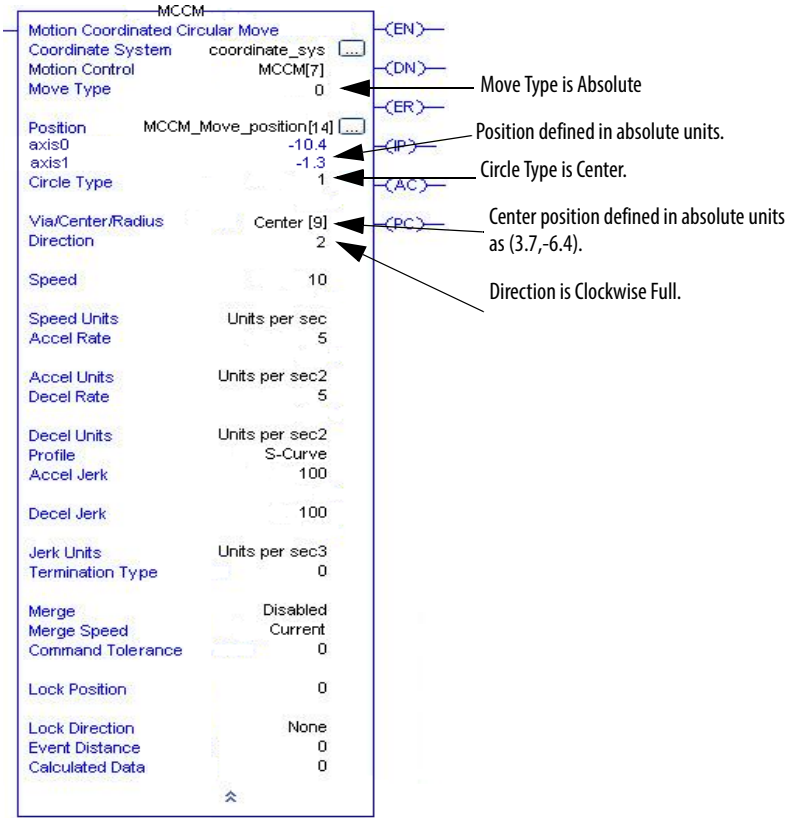
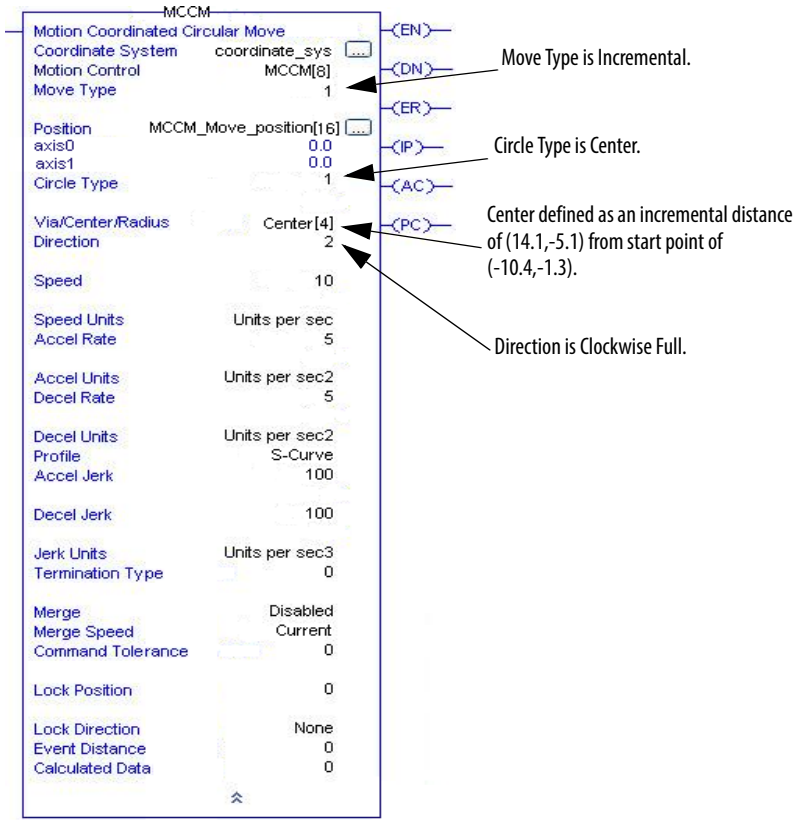


Figure 79 - MCCM with Move Type as Incremental and Center Type as Center



To draw a full circle by using Radius as the Circle Type:

- The starting point must not equal the end point.
- The direction must be either Clockwise Full or Counter Clockwise Full.
- The sign of Radius is irrelevant.

MCCM with Rotary Axes Examples

The following examples show the use of the MCCM instruction with Rotary axes and Move Types of Absolute and Incremental.

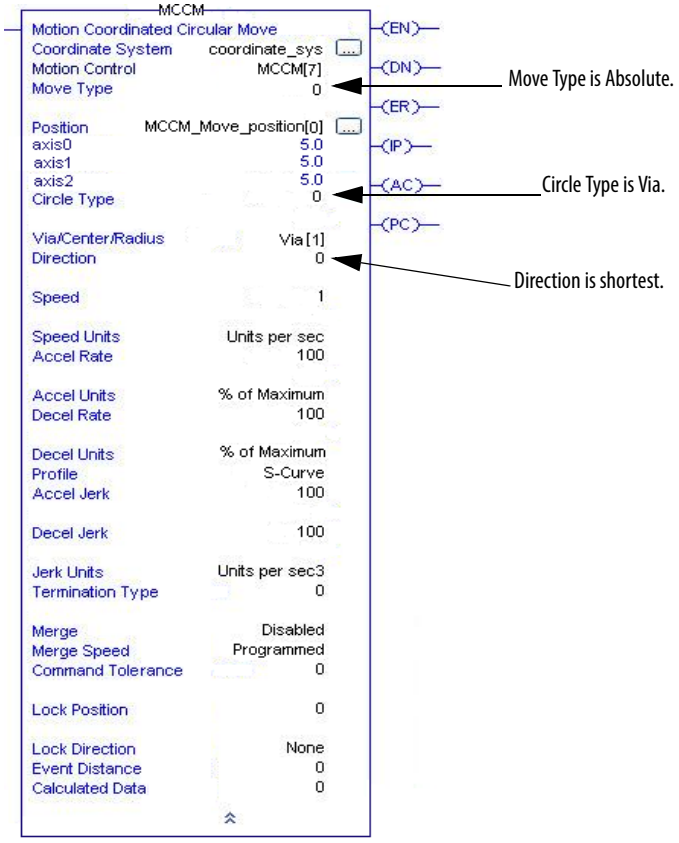
MCCM Instruction with Three Axes, One Rotary Axis, and Move Type of Absolute

The first example uses a coordinate system of three axes with one Rotary axis and a Move type of Absolute. The plot of the path is based on the following assumptions:

- Three-axis Coordinate System named coordinate_sys (Axis2, the Z axis, is ignored in plots to reduce the confusion and to better illustrate the actions of the rotary axis (Axis0).
- Axis0 is Rotary with an unwind of 5 revs.

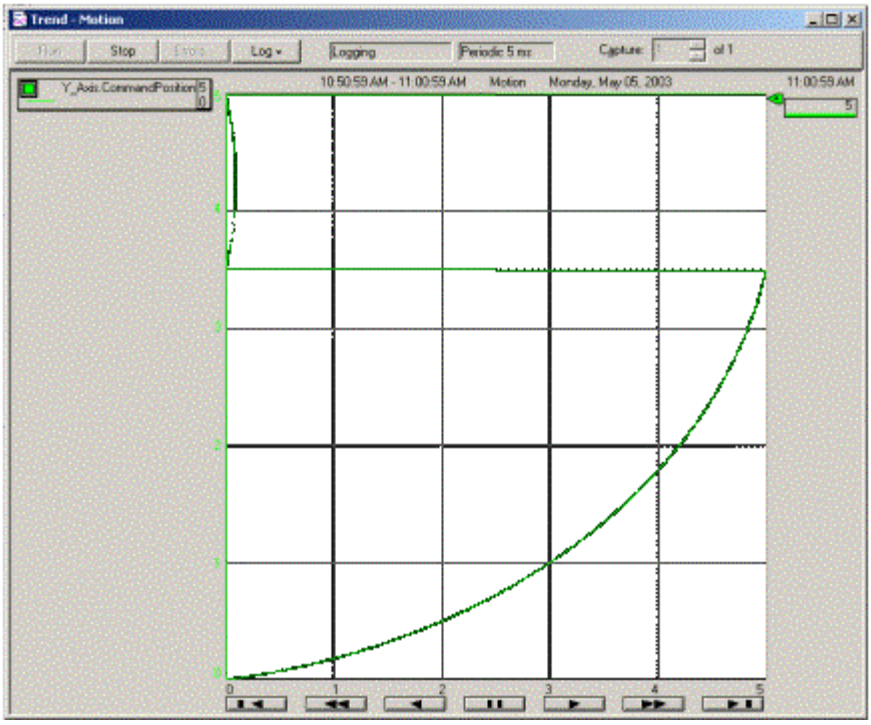
- Start position is 0, 0, 0.
- End position is 5, 5, 5.
- Via position is 5, 3.5, 3.5.

Figure 80 - MCCM Ladder Instruction with Move Type of Absolute



The preceding MCCM instruction produces the following plot.

Figure 81 - Plot of MCCM with Three Axes, One Rotary Axis & Move Type of Absolute



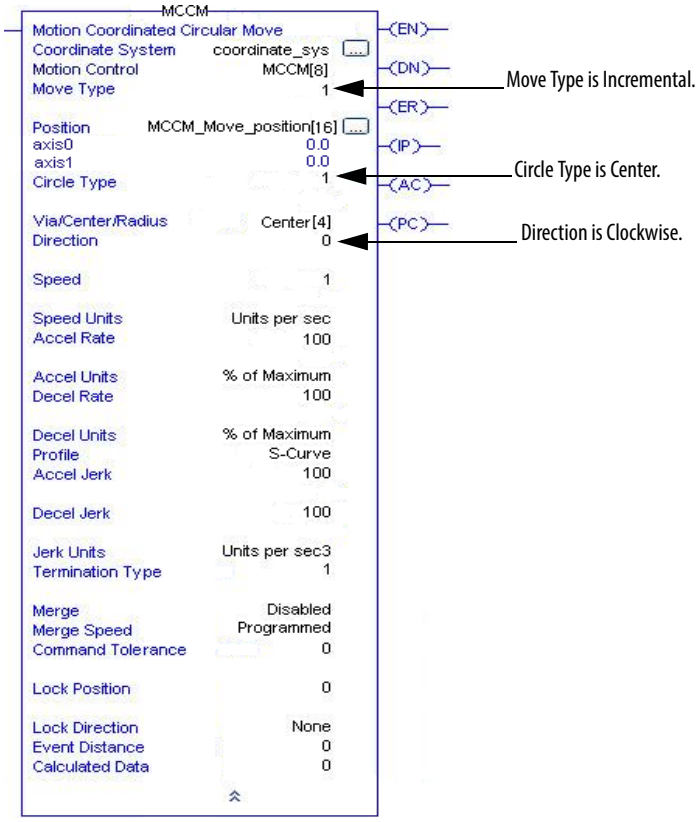
The axis actually travels counter clockwise in an arc from (0,0,0) to (5,5,5) via the (5,3.5,3.5) position. The Direction was specified as clockwise but with Via specified for the Circle Type, the Direction operand is ignored. The move stops after generating a 90 degree arc. There was one travel through the unwind for Axis0 even though it was in Move Type of Absolute. Note that the path of the coordinated motion is determined in linear space but the position of the axes is limited by the rotary configuration. The End and Via points are required to fit within the absolute position defined by the rotary unwind of Axis0. However, the resulting motion from these choices can travel through the unwind of the rotary axis.

MCCM Instruction with Two Rotary Axis and Move Type of Incremental

This example uses a coordinate system of two Rotary axes and a Move type of Incremental. The plot of the path is based on these assumptions.

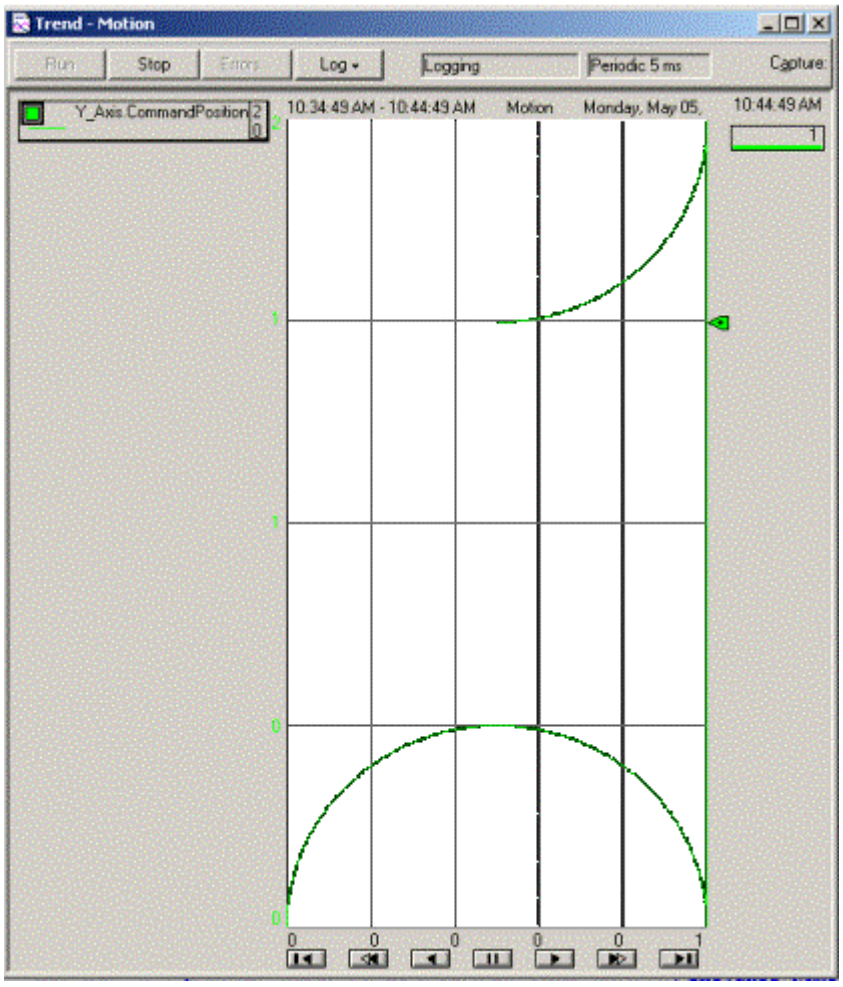
- Two-axis coordinate system named coordinate_sys.
- Axis0 is Rotary with an unwind of 1 rev.
- Axis1 is Rotary with an unwind of 2 revs.
- Start position is 0, 0.
- Increment to end position is 0.5, -0.5.
- Increment to Center position is 0.5, 0.

Figure 82 - MCCM Ladder Instruction with Move Type of Absolute



The preceding MCCM instruction produces the following plot.

Figure 83 - Plot of MCCM with Two Rotary Axes and Move Type of Incremental



The axis travels clockwise in a circle from (0,0) to (0.5,1.5). The move stops after generating a 270 degree arc. There was one travel through the unwind for Axis1. Note that the path of the coordinated motion is determined in linear space but the position of the axes is limited by the rotary configuration. The endpoint was (0.5,-0.5) for the circle calculations but the actual endpoint for the move was (0.5,1.5). The instruction specified and we obtained a clockwise move even though one axis had a negative incremental target position. The endpoint is not required to fit within the absolute position defined by the rotary unwind of the axes.

Three-dimensional Arc Examples

For Coordinate Systems that have three primary axes associated to them, it is possible to create three-dimensional arcs.

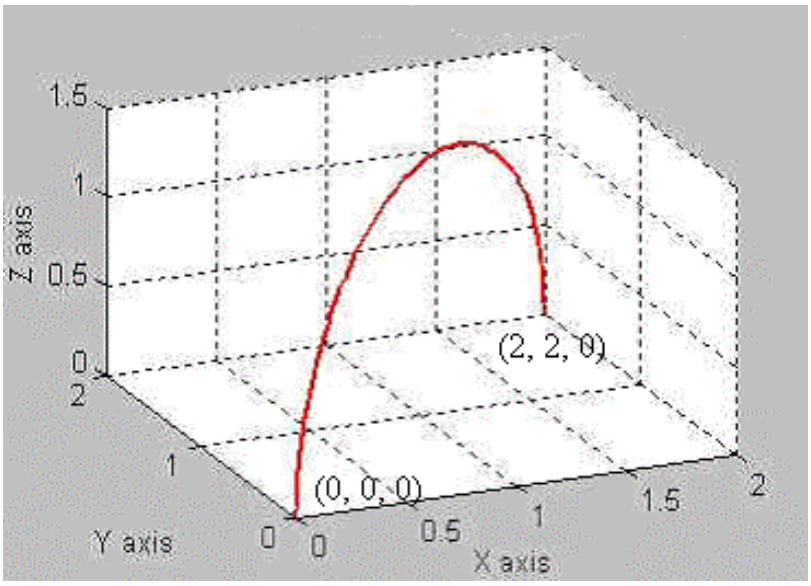
MCCM Using Via Circle Type

The following example shows the use of the MCCM with a Circle Type of Via and a Move Type of Absolute to create a three-dimensional arc. The basic assumptions are:

- the three axes, Axis0 and Axis1, Axis2 are all members of the coordinate system, coordinate_sys.
- coordinate_sys is a three-dimensional coordinate system.
- Axis0, Axis1, and Axis2 are orthogonal to each other.
- coordinate_sys is initially at (0.0, 0.0, 0.0) units.

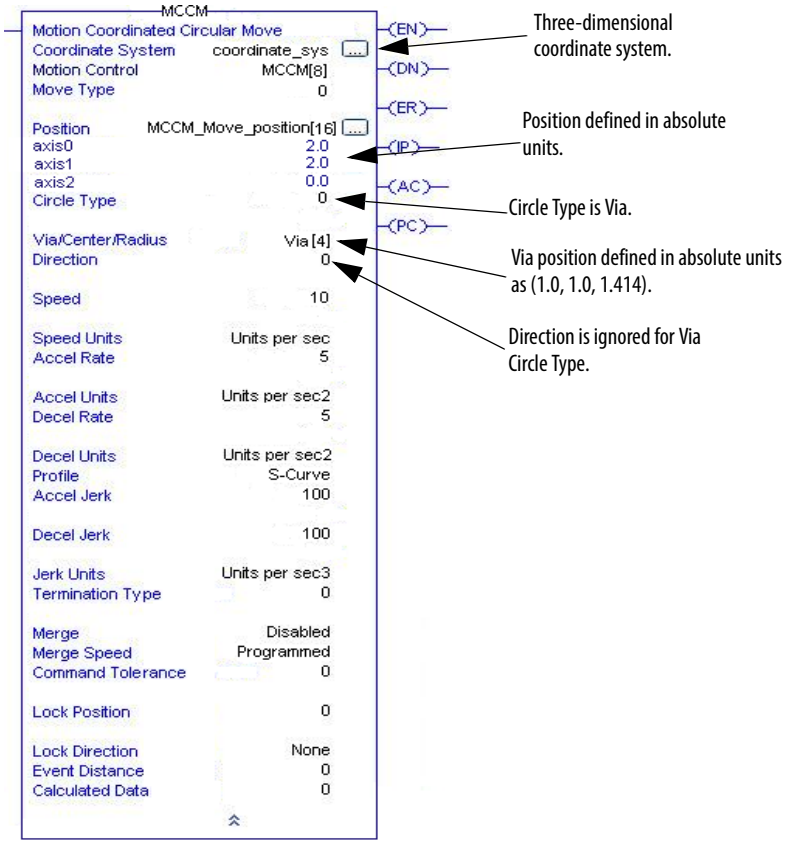
Move Coordinated_sys1 along an arc to (2.0, 2.0, 0.0) units passing through (1.0, 1.0, 1.414) units at the vector speed of 10.0 units per second with the acceleration and deceleration values of 5.0 units per second^2. The following graph shows the 3D arc generated by the preceding information.

Figure 84 - Three-dimensional Arc Using Circle Type of Via



This path is achieved by using an MCCM instruction with a Move Type of Absolute and a Circle Type of Via. When Via is selected, the Via/Center/Radius position defines a point through which the arc must pass.

Figure 85 - MCCM Ladder Instruction for 3D Arc Using Circle Type of Via



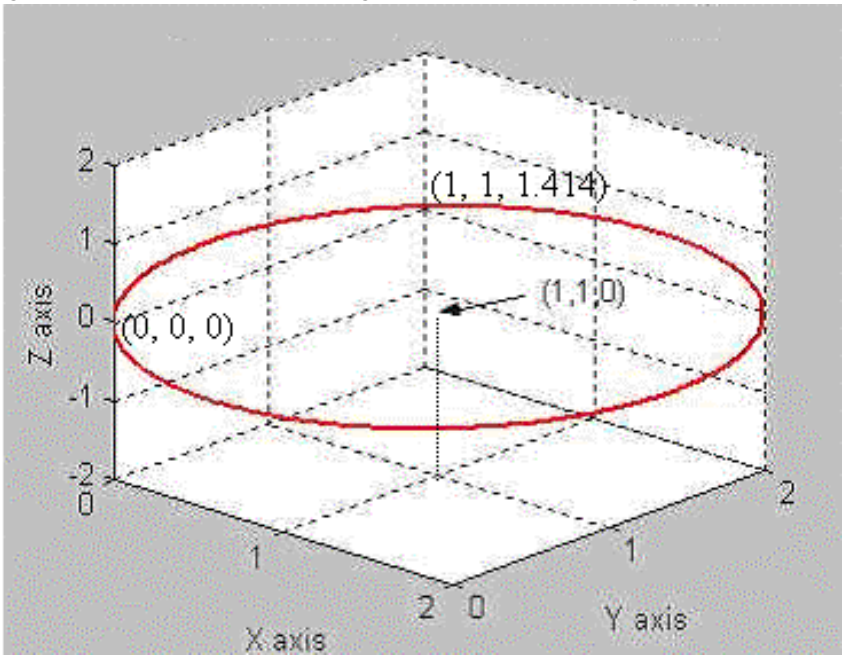
MCCM Using Center Circle Type

The following example shows the use of the MCCM with a Circle Type of Center and a Move Type of Absolute to create a three-dimensional arc. The basic assumptions are:

- the three axes, Axis0 and Axis1, Axis2 are all members of the coordinate system, coordinate_sys.
- coordinate_sys is a three-dimensional coordinate system.
- Axis0, Axis1, and Axis2 are orthogonal to each other.
- coordinate_sys is initially set at (0.0, 0.0, 0.0) units.

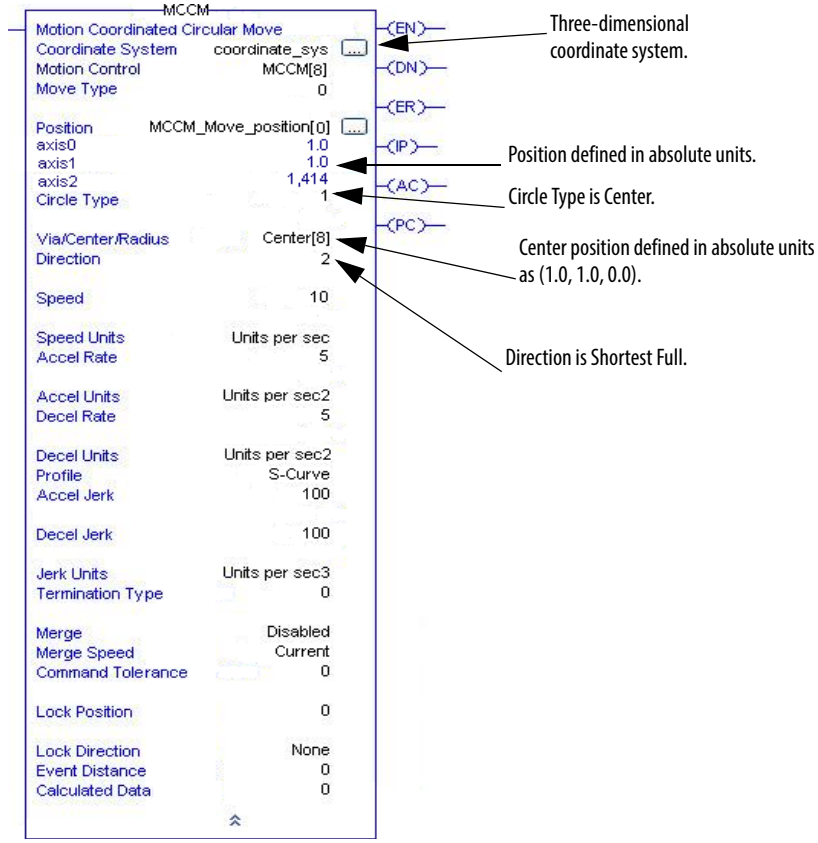
Move Coordinated_sys1 along an arc to (1.0, 1.0, 1.414 units with center at (1.0, 1.0, 1.0) units at the vector speed of 10.0 units per second with the acceleration and deceleration values of 5.0 units per second^2. The following graph shows the three-dimensional arc generated by the preceding information.

Figure 86 - Three-dimensional Path Using Shortest Full for Direction Operand



This path is achieved by using an MCCM instruction with a Move Type of Absolute and a Circle Type of Center. When Via is selected, the Via/Center/Radius position defines a point through which the arc must pass.

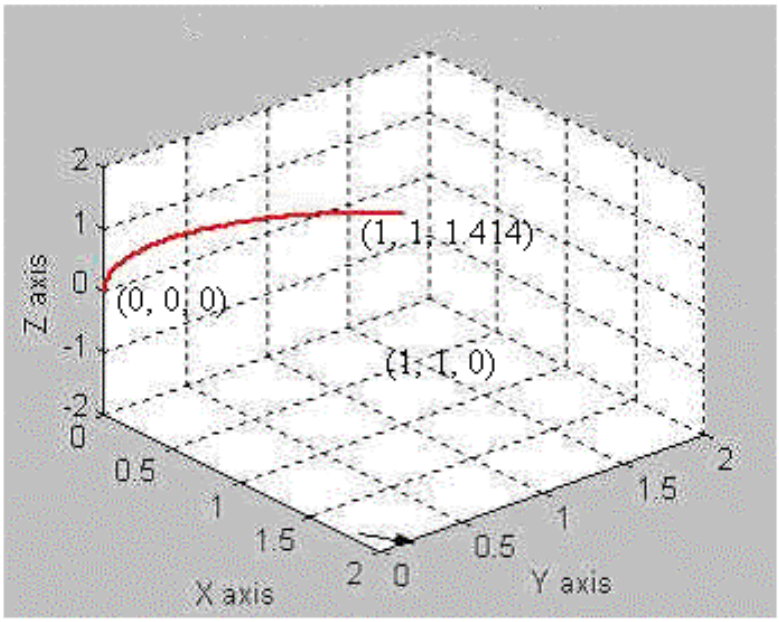
Figure 87 - MCCM Ladder Instruction for 3D Arc Using Circle Type of Center



For full circles, set Position operand to any point except the start point and use one of the Full Direction types. The endpoint is assumed to be the start point. This is because in the three-dimensional space, you need three points to specify a plane for the circle.

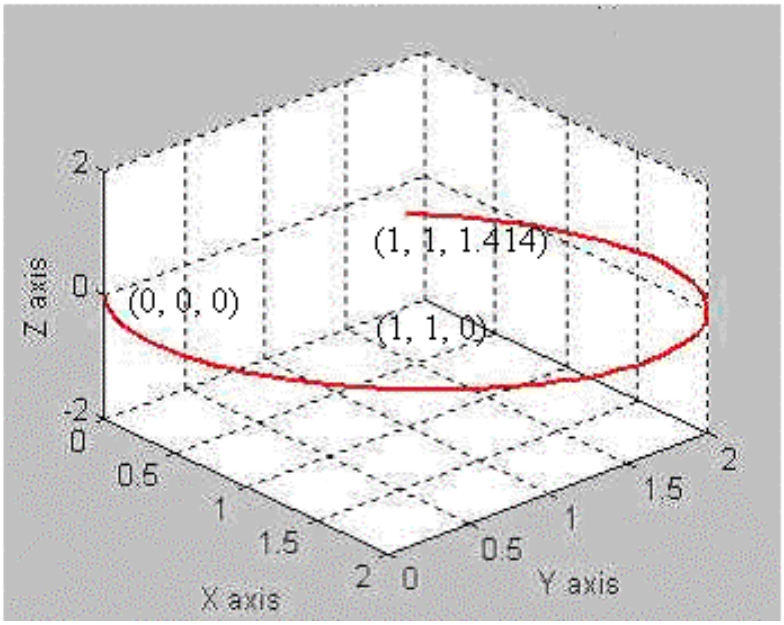
By changing the Direction operand to Shortest in the preceding MCCM instruction, the following path is generated. The Shortest option of the Direction operand takes the shortest route from the start point to the point defined by the Position operand of the MCCM instruction.

Figure 88 - 3D Path Using Shortest for Direction Operand



Change the Direction operand to Longest in the preceding MCCM instruction and the path followed is the longest from the start point to the point defined by the Position operand in the MCCM instruction. See the following diagram for an example of the longest path.

Figure 89 - 3D Path Using Longest for Direction Operand



Calculate Jerk Units

The jerk units define the units that are applied to the values entered in the Accel Jerk and Decel Jerk operands. The values are entered directly in the position units of the specified coordinate system or as a percentage. When configured by using % of Maximum, the jerk is applied as a percentage of the Maximum Acceleration Jerk and Maximum Deceleration Jerk operands specified in the coordinate system attributes. When configured by using % of Time, the value is a percentage based on the Speed, Accel Rate, and Decel Rate specified in the instruction.

Convert Engineering Units to a Percentage of Time

If you want to convert engineering units to % of Time, use these equations.

For Accel Jerk:

$$j_a\,[\mathrm{EU/s^3}] = \frac{{a_{max}}^2[\mathrm{EU/s^2}]}{v_{max}\,[\mathrm{EU/s}]}\left(\frac{200}{j_a\,[\%\ \mathrm{of\ time}]} - 1\right)$$

For Decel Jerk:

$$j_a\,[\mathrm{EU/s^3}] = \frac{{a_{max}}^2[\mathrm{EU/s^2}]}{v_{max}\,[\mathrm{EU/s}]}\left(\frac{200}{j_a\,[\%\ \mathrm{of\ time}]} - 1\right)$$

Convert Percentage of Time to Engineering Units

If you want to convert % of Time to engineering units, use these equations.

For Accel Jerk:

$$j_a\,[\%\ \mathrm{of\ time}] = \frac{2}{1 + \frac{j_a\,[\mathrm{EU/s^3}]\ v_{max}\,[\mathrm{EU/s}]}{a_{max}^2\,[\mathrm{EU/s^2}]}}\,100$$

For Decel Jerk:

$$j_d\,[\%\ \mathrm{of\ time}] = \frac{2}{1 + \frac{j_d\,[\mathrm{EU/s^3}]\ v_{max}\,[\mathrm{EU/s}]}{d_{max}^2\,[\mathrm{EU/s^2}]}}\,100$$

Important Consideration

If you program tangent circles with different Jerk rates (Decel Jerk of first circle and Accel Jerk of the second circle), then you can get a slight velocity discontinuity at the intersection of the two circles. The size of the discontinuity depends on the magnitude of the Jerk difference. In other words, the smaller the Jerk difference, the smaller the velocity glitch. Therefore, we recommend that you do not program Jerk rates on tangent circles.

Merge

The merge defines whether or not to turn the motion of all specified axes into a pure coordinated move.

Table 30 - Merge Options

Option	Description
Merge Disabled	Any currently executing single axis motion instructions involving any axes defined in the specified coordinate system are not affected by the activation of this instruction, and result in superimposed motion on the affected axes. An error is flagged if a second instruction is initiated in the same coordinate system or in another coordinate system containing any axes in common with the coordinate system that is active.
Coordinated Motion	Any currently executing coordinated motion instructions involving the same specified coordinate system are terminated, and the active motion is blended into the current move at the speed defined in the merge speed parameter. Any pending coordinated motion instructions in the specified coordinate system are cancelled. Any currently executing system single axis motion instructions involving any axes defined in the specified coordinate system are not affected by the activation of this instruction, and result in superimposed motion on the affected axes.
All Motion	Any currently executing single axis motion instructions involving any axes defined in the specified coordinate system and any currently executing coordinated motion instructions are terminated. The prior motion is merged into the current move at the speed defined in Merge Speed parameter. Any pending coordinated move instructions are cancelled.

Programming Guidelines for Zero Length Moves

In Master Driven Mode and Time Driven Mode, you have the option of configuring a move with a Slave distance increment of zero (or a move with the same target and current position). If speed is specified in Master Units, the move remains active until the specified Master distance has been traversed. Use this type of move to generate a dwell in a multi-segment path move.

Similarly, when you program the move time directly in seconds in Time Driven Mode, a move of the duration of X seconds with a zero departure results in a programmed delay of the specified time.

IMPORTANT Instructions with zero length cause velocity of the multi-axis instruction preceding the one with zero length to decelerate to zero at its endpoint. To avoid this behavior, it is suggested that you eliminated moves with zero length from your program.

A zero length move with a duration of zero time completes in the minimum time possible, which is 1 coarse iteration.

Dwells

You have the option to program a dwell by using Time Based Programming in either Time Driven Mode or MDSC Mode when a zero length move (see Zero Length Move below) is programmed. The acceleration, deceleration, and jerk parameters are ignored when a zero length move is programmed. Therefore,

when in time driven mode, the duration of the dwell is in seconds. When in MDSC mode, the duration of the dwell is programmed in units of Master Distance.

In MDSC mode, the dwell starts either at the Master Lock Position or immediately, depending on the programmed Lock Direction parameter, and continues for a duration as specified in the Speed parameter.

Time Based Programming Errors

- A zero length move with a duration of zero time completes in 1 coarse iteration, which is the minimum time possible.
- A zero length move that is programmed with Speed Units other than seconds or master distance completes almost immediately.
- An error occurs if a move is programmed by using Time Based Planning that is started with a nonzero velocity. This means that a move using the merge enabled parameter in an instruction causes an error for most cases because merge is typically used when the axes are moving.
- An error occurs if speed is programmed in units of seconds and acceleration, deceleration, or jerk is not programmed in seconds (or % of Time for jerk).

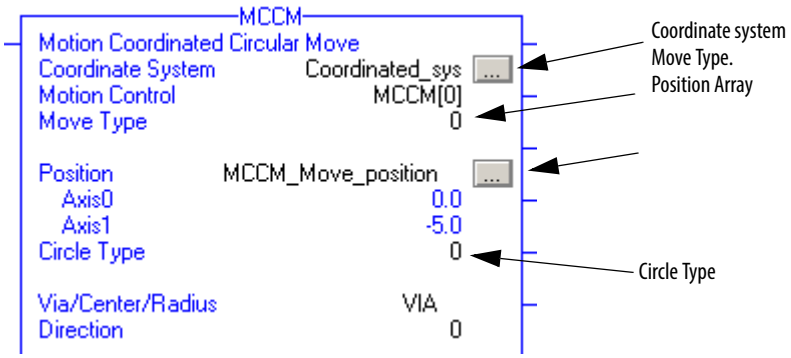
MCCM Target Position Entry Dialog Box

The MCCM Target Position Entry Dialog box is accessed by pressing the ellipsis button to the right of the position operand of the ladder instruction faceplate. The Target Position Entry box can only be accessed if the coordinate system for the instruction has:

- been named,
- a valid tag name for the Position operand that contains enough elements to accommodate the number of axes,
- selected a valid Move Type and a valid Circle Type.

If these criteria have not been satisfied, an error message is displayed on the status bar

Figure 90 - MCCM Ladder Valid Values for Accessing Target Position Entry Box.



Press the ellipsis and the following dialog box appears.

Figure 91 - MCCM Instruction Target Position Entry Dialog Box - Position Tab

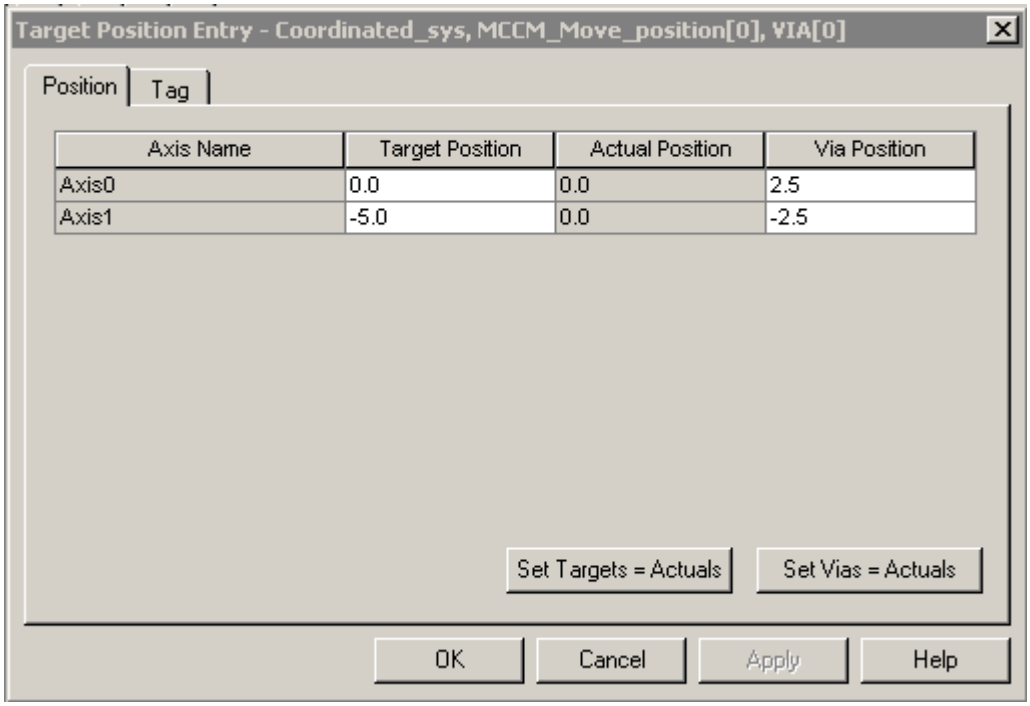


Table 31 - Target Position Entry Dialog Box Fields

Feature	Description
Axis Name	This column has the names of each axis in the coordinate system named in the ladder faceplate. You cannot change these names.
Target Position/Target Increment	The values in this column are numeric. They show the endpoint or incremental departure of the move depending on the active Move Type. The column heading indicates which is displayed.
Actual Position	This column contains the current actual position of the axes in the coordinate system. These values update dynamically when on-line and the Coordinate System Auto Tag Update is enabled.
Via Position/Via Increment Center Position/Center Increment Radius	Depending on the Circle Type selected, this column contains the Via point position or increment, the Center Position or increment.
Set Targets = Actuals	This button is enabled when the Move Type is Absolute and is used to copy the value from the Actual Position fields to the Target Position fields.
Set Vias = Actuals	This button is only active if the Move Type is Absolute. It is used to copy the values from the Actual Position fields to the Vias Fields.

The Move Type and Circle Type selected govern the appearance of this dialog box. The following table illustrates how the screen is affected by the combinations of Move Type and Circle Type selected.

Table 32 - Target Position Entry Dialog Box Changes

Move Type	Circle Type	Behavior
Absolute	Via	Target column is entitled Target Position. Via column is entitled Via Position. Set Targets = Actuals button is active. Set Vias = Actuals button is active.
Incremental	Via	Target column is entitled Target Increment. Via Column is entitled Via Increment. Set Targets = Actuals button is inactive (Grayed Out). Set Vias = Actuals button is inactive (Grayed Out).
Absolute	Center	Target column is entitled Target Position. Center column is entitled Center Position. Set Targets = Actuals button is active. Set Vias = Actuals button is active.
Incremental	Center	Target column is entitled Target Increment. Center Column is entitled Center Increment. Set Targets = Actuals button is inactive (Grayed Out). Set Vias = Actuals button is inactive (Grayed Out).
Absolute	Radius	Target column is entitled Target Position. Radius column is entitled Radius. Set Targets = Actuals button is active. Set Vias = Actuals button is inactive (Grayed Out).
Incremental	Radius	Target column is entitled Target Increment. Radius Column is entitled Radius. Set Targets = Actuals button is inactive (Grayed Out). Set Vias = Actuals button is inactive (Grayed Out).
Absolute	Center Incremental	Target column is entitled Target Position. Center Incremental column is entitled Center Incremental. Set Targets = Actuals button is active. Set Vias = Actuals button is inactive (Grayed Out).
Incremental	Center Incremental	Target column is entitled Target Increment. Center Incremental column is entitled Center Incremental. Set Targets = Actuals button is inactive (Grayed Out). Set Vias = Actuals button is inactive (Grayed Out).

Arithmetic Status Flags

Not affected.

Fault Conditions

None.

Error Codes

See Error Codes (ERR) for Coordinate Motion Instructions on page 261.

Runtime Error Conditions

- You cannot switch from Time Driven Mode to Master Driven Mode if the master speed is zero unless the slave speed is zero too.
- The slave move must start at rest if Speed Units = Seconds or Master Units. Any of the following conditions can cause this error:
  - MCCM with Merge = Coordinated Motion or Merge = All Motion and Speed = Seconds or Master Units is started while another MCCM is in progress.

MCCM uses Term Type = 4 or 5 and Speed = Seconds or Master Units.

Extended Error Codes

Extended Error codes help to further define the error message given for this particular instruction. Their behavior is dependent upon the Error Code with which they are associated.

The Extended Error Codes for Servo Off State (5), Shutdown State (7), Axis Type Not Servo (8), Axis Not Configured (11), Homing In Process Error (16), and Illegal Axis Data type (38) errors all function in the same fashion. A number between 0...*n* is displayed for the Extended Error Code. This number is the index to the Coordinate System indicating the axis that is in the error condition.

For Error Code Axis Not Configured (11) there is an additional value of -1 that indicates that Coordinate System was unable to setup the axis for coordinate motion.

For the MCCM instruction, Error Code 13 - Parameter Out of Range, Extended Errors returns a number that indicates the offending parameter as listed on the faceplate in numerical order from top to bottom beginning with zero. For example, 2 indicates the parameter value for Move Type is in error.

Table 33 - Error Code 13 Extended Errors

Error Code and (Number)	Extended Error Numeric Indicator	Instruction Parameter	Description
Parameter Out Of Range (13)	0	Coordinate System	Number of primary axes is not 2 or 3.
Parameter Out Of Range (13)	2	Move Type	Move Type is either less than 0 or greater than 1.
Parameter Out Of Range (13)	3	Position	The position array is not large enough to provide positions for all the axes in the coordinate system.
Parameter Out Of Range (13)	4	Circle Type	Circle Type is either less than 0 or greater than 4.
Parameter Out Of Range (13)	5	Via/Center/Radius	The size of the Via/Center array is not large enough to provide positions for all of the axes in the defining via/center point.
Parameter Out Of Range (13)	6	Direction	Direction is either less than 0 or greater than 3.
Parameter Out Of Range (13)	7	Speed	Speed is less than 0.
Parameter Out Of Range (13)	9	Accel Rate	Accel Rate is less than or equal to 0.
Parameter Out Of Range (13)	11	Decel Rate	Decel Rate is less than or equal to 0.
Parameter Out Of Range (13)	14	Termination Type	Termination Type is less than 0 or greater than 3.

Error Code 54 – Maximum Deceleration Value is Zero

If the Extended Error returns a positive number (0-*n*) it's referring to the offending axis in the coordinate system.

1. Go to the Coordinate System Properties General Tab and look under the Brackets ([])column of the Axis Grid to determine which axis has a Maximum Deceleration value of 0.
2. Click the ellipsis next to the offending axis to access the Axis Properties screen.
3. Go to the Dynamics tab and make the appropriate change to the Maximum Deceleration Value.

   If the Extended Error number is -1, this means the Coordinate System has a Maximum Deceleration Value of 0.
4. Go to the Coordinate System Properties Dynamics Tab to correct the Maximum Deceleration value.

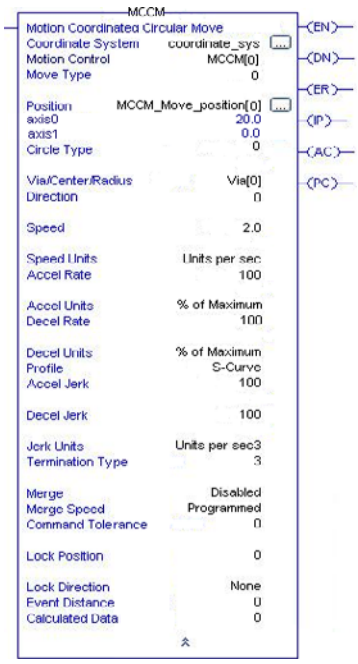
Circular Error Examples

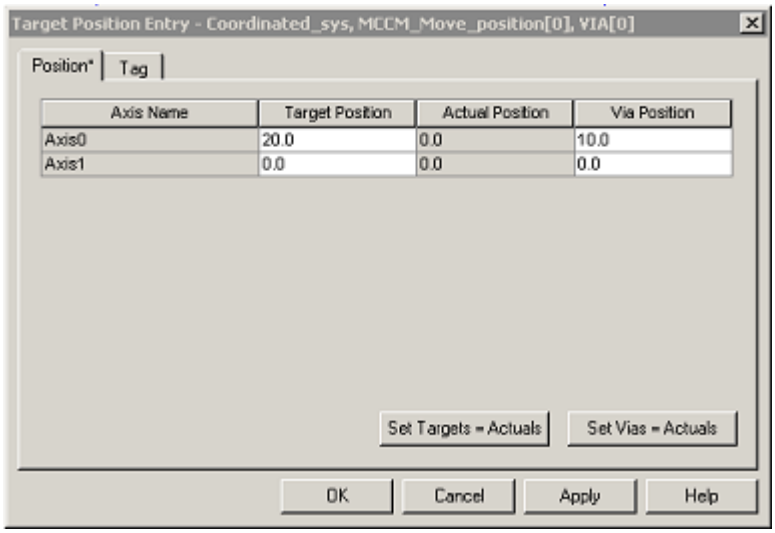
Due to the complexity of the MCCM instruction and the error codes it can generate, the following simple examples are given to aide in the understanding of the MCCM instruction.

CIRCULAR_COLLINEARITY_ERROR (44) Example

The following example for error #44 shows a situation where the startpoint, via-point, and endpoint all lie on a straight line. The program is trying to generate a two dimensional arc going from 0,0 (current position) to 20,0 through the location 10,0. Because these points all lie on a straight line, no circular centerpoint can be computed for the circle. This error is also generated if the program was for a three-dimensional center type circle using a startpoint, centerpoint, and endpoint all lying on a straight line. Here, an infinite number of circles could fit through the programmed points in an infinite number of planes.

Figure 92 - Ladder Program and Target Entry Screen that Generate Error #44.

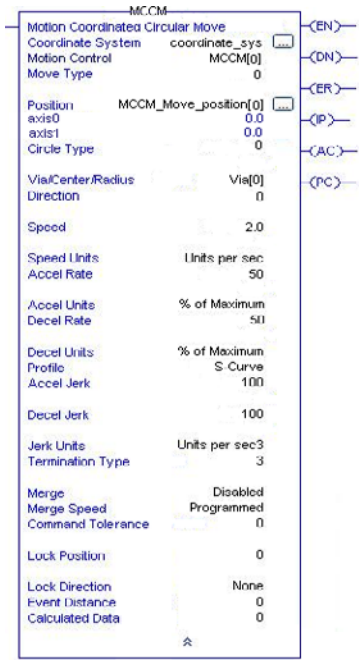


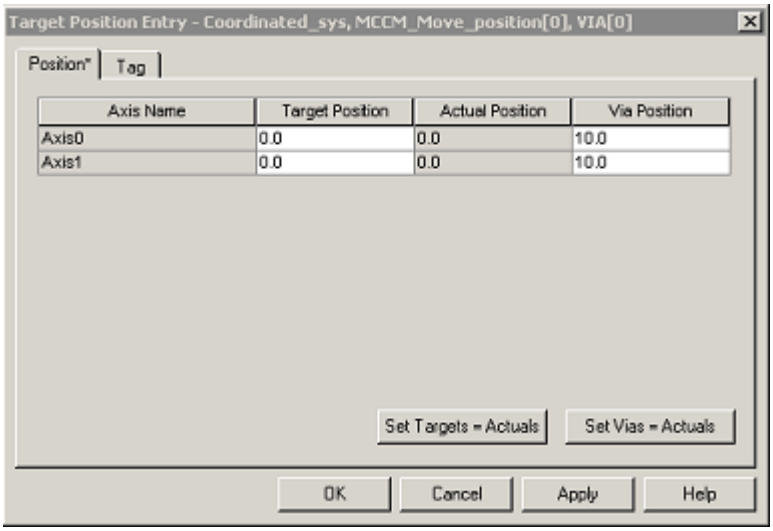


CIRCULAR_START_END_ERROR (45) Example

The following example for error #45 depicts a situation where the startpoint and via-point are the same. The program is trying to generate a two dimensional full circle from 0,0 (current position) back to 0,0 through the location 10,10. Because the startpoint and the via-point are the same, no circular centerpoint can be found for this circle.

Figure 93 - Ladder Program and Target Entry Screen that Generate Error #45

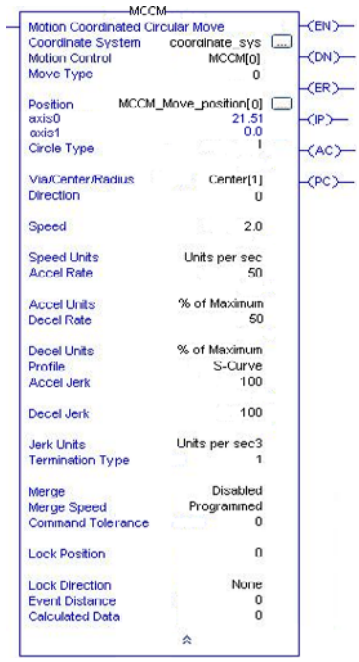


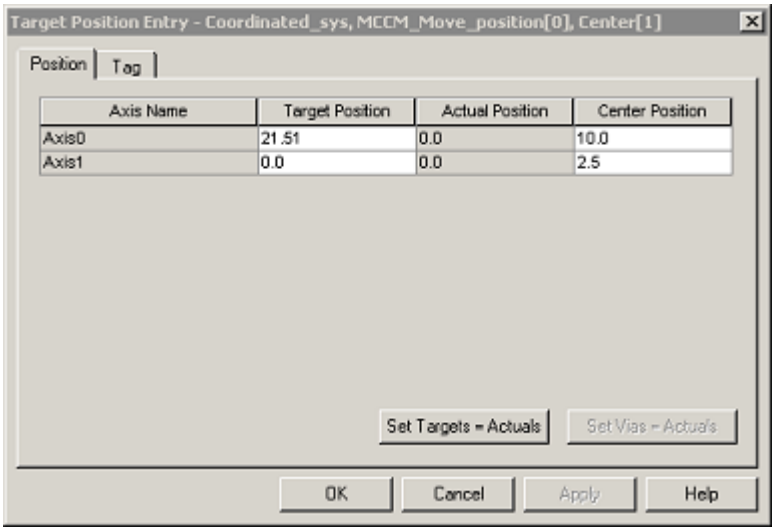


CIRCULAR_R1_R2_MISMATCH_ERROR (46) Example

The following example for error #46 shows a situation where the difference in radial start/end lengths exceeds 15% of the radial start length. The program is trying to generate a two dimensional arc from 0,0 (current position) to 21.51,0 by using a centerpoint at 10,10. Because the difference of the radial start/end lengths is 21.51 - 10 = 1.51, it exceeds 15% of the radial start length .15 * 10 = 1.5. This example works with an endpoint of 21.5 and the centerpoint recomputed to lie exactly halfway between start and end points.

Figure 94 - Ladder Program and Target Entry Screen that Generate Error #46

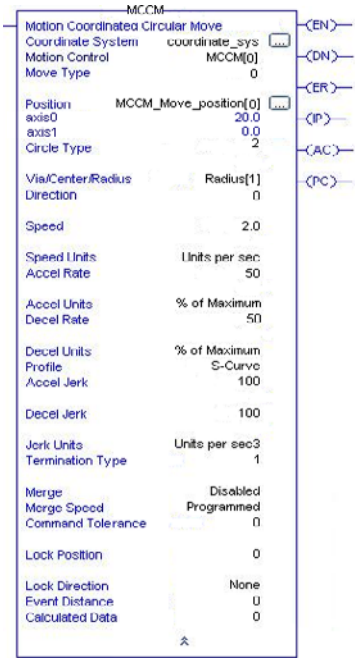


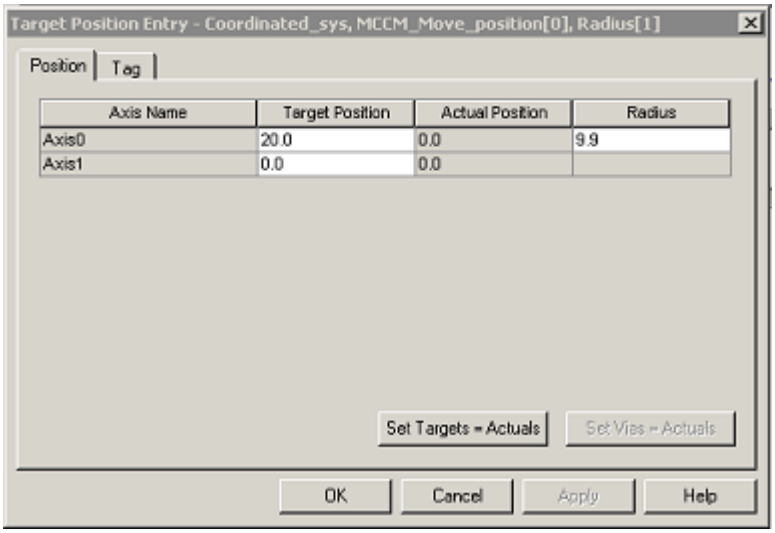


CIRCULAR_SMALL_R_ERROR (49) Example

This first example of error #49 depicts a situation where the radius type circle uses a radius that is too short to span the distance between the start point and the end point. The program is trying to generate a two dimensional arc going from 0,0 (current position) to 20,0. However, the programmer tried to program a radius type circle with a radius that is too short to span the distance between the startpoint and endpoint.

Figure 95 - Ladder Program and Target Entry Screen that Generate Error #49

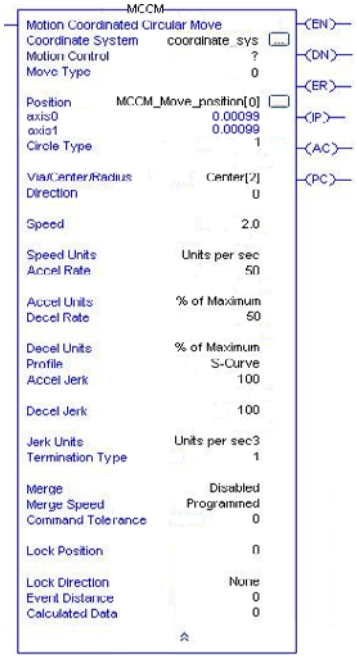


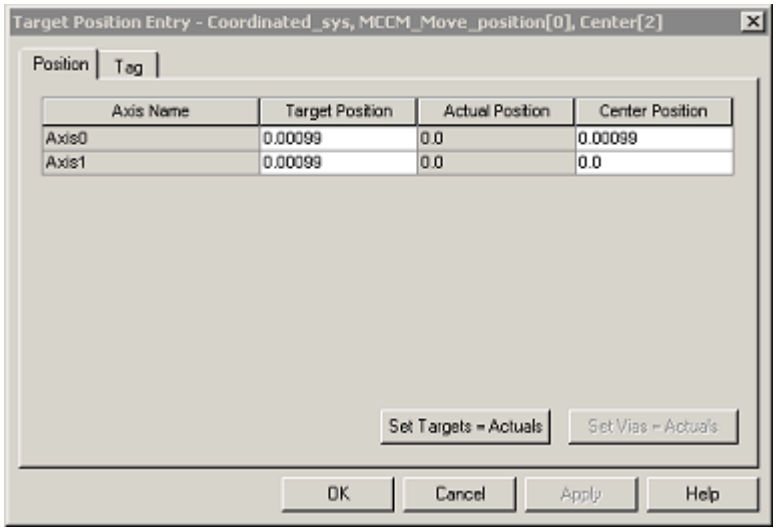


CIRCULAR_SMALL_R_ERROR (49) Example

This second example of error #49 shows a situation where the radius type circle uses a radius of magnitude of less than 0.001. The program is trying to generate a two dimensional arc going from 0,0 (current position) to 0.00099,0.00099. This error occurs because the programmer tried to program a radius type circle with a radius of a magnitude less than 0.001 units.

Figure 96 - Ladder Program and Target Entry Screen that Generate Error #49





Circular Programming Reference Guide

Circle Type	Used in 2D/3D/Both	Validation Errors	Direction – 2D	Direction – 3D	Comments
Radius	2D	Error 25; Illegal Instruction Error 45 Endpoint = Startpoint Error 49; R too small (|R| < .001) or R too short to span programmed points.	CW/CCW as viewed from the '+' perpendicular to the circular plane.	N/A	A '+" radius forces arc length to be <= 180° (Shortest arc). A "-" radius forces arc length to be => 180° (Longest arc). Full Circles can be programmed. For full circles, set Position to be any point on circle except Startpoint and use one of the Full direction types.
Center Point	Both	Error 44; Collinearity (3D only) Error 45; Endpoint = Startpoint (3D only) Error 46; Start/End radius mismatch (|R1 - R2| > .15 * R1).	CW/CCW as viewed from the '+' perpendicular to the circular plane.	Shortest/Longest arc. In Full circles, placement of endpoint defines shortest/longest paths referred to by direction parameter.	1. Full Circles can be programmed. 2. In 2D only, Endpoint = Startpoint is legal. Therefore, full circles can be generated: – By setting Endpoint = Startpoint, in which case, all direction types produce full circles. – By setting Endpoint not = Startpoint and using Full direction type. 3. For 3D Full Circles, set Position to be any point on the circle except Startpoint, and use one of the Full direction types. Position defines both arc and Shortest direction types.
Via Point	Both	Error 44; Collinearity Error 45; Endpoint = Startpoint	Via point always determines direction.	Via point always determines direction. Direction operand is only used to determine if circle is partial or full.	1. Full Circles can be programmed. 2. For full circles, set Position to be any point on circle except Startpoint and use one of the Full direction types.

MCCM Changes to Status Bits

Status bits provide a means for monitoring the progress of the motion instruction. There are three types of Status bits that provide pertinent information.

- Axis
- Coordinate System
- Coordinate Motion

When the MCCM instruction initiates, the status bits undergo the following changes.

Table 34 - Axis Status Bits

Bit Name	Meaning
CoordinatedMotionStatus	Sets when the MCCM instruction executes and is cleared when the instruction completes.

Table 35 - Coordinate System Status Bits

Bit Name	Meaning
MotionStatus	Sets when the MCCM instruction is active and the Coordinate System is connected to its associated axes.

Table 36 - Coordinate Motion Status Bits

Bit Name	Meaning
AccelStatus	Sets when vector is accelerating. Clears when a blend is in process or when vector move is at speed or decelerating.
DecelStatus	Sets when vector is decelerating. Clears when a blend is in process or when vector move is accelerating or when move completes.
ActualPosToleranceStatus	Sets for Actual Tolerance termination type only. The bit is set after the following two conditions have been met. 1) Interpolation is complete. 2) The actual distance to the programmed endpoint is less than the configured coordinate system's Actual Tolerance value. It remains set after the instruction completes. It is reset when a new instruction is started.
CommandPosToleranceStatus	Sets for all termination types whenever the distance to the programmed endpoint is less than the configured coordinate system's Command Tolerance value and remains set after the instruction completes. It is reset when a new instruction is started. The CommandPosToleranceStatus (CS_CMD_POS_TOL_STS) status bit in the Coordinate System is set as follows: TT0, TT1, TT4, TT5 - Bit is set when the distance to the endpoint is less than the Command Tolerance value. The bit is cleared when the first move is complete. TT2, TT6 - Bit is set when the distance to the endpoint is less than the Command Tolerance value. The bit is cleared when the blend is started (that is, when the second move is started). Thus, if the blend is started at the Command Tolerance (CT) point, the bit is not shown. If the next move is a short move or for time matching of the acceleration and deceleration of the two adjacent moves, it can result in the blend being deferred slightly beyond the CT point. TT3 - Bit is set when the distance to the endpoint is less than the Command Tolerance value (like TT2 and TT6). The bit is cleared when the blend is started. Thus, the bit does not display if the blend is started at the deceleration point. If the next move is a short move or for time matching of the acceleration and deceleration of the two adjacent moves, it can result in the blend being deferred slightly beyond the CT point.
StoppingStatus	The Stopping Status bit is cleared when the MCCM instruction executes.
MoveStatus	Sets when MCCM begins axis motion. Clears on the .PC bit of the last motion instruction or a motion instruction executes, which causes a stop.
MoveTransitionStatus	Sets when No Decel or Command Tolerance termination type is satisfied. When blending collinear moves, the bit is not set because the machine is always on path. It clears when a blend completes, the motion of a pending instruction starts, or a motion instruction executes, which causes a stop. Indicates not on path.
MovePendingStatus	Sets when one pending coordinated motion instruction is in the instruction queue. Clears when the instruction queue is empty.
MovePendingQueueFullStatus	Sets when the instruction queue is full. It clears when the queue has room to hold another new coordinated move instruction.

Table 36 - Coordinate Motion Status Bits

Bit Name	Meaning
CoorMotionLockStatus	Set when an axis lock is requested for an MCLM or MCCM instruction and the axis has crossed the Lock Position. Cleared when an MCLM or MCCM is initiated. For the enumerations Immediate Forward Only and Immediate Reverse Only, the bit is set immediately when the MCLM or MCCM is initiated. When the enumeration is Position Forward Only or Position Reverse Only, the bit is set when the Master Axis crosses the Lock Position in the specified direction. The bit is never set if the enumeration is NONE. The CoordMotionLockStatus bit is cleared when the Master Axis reverses direction and the Slave Axis stops following the Master Axis. The CoordMotionLockStatus bit is set again when the Slave Coordinate System resumes following the Master Axis. The CoordMotionLockStatus bit is also cleared when an MCCS is initiated.

Coordinated Motion only supports the queueing of one coordinated motion instruction. Therefore the MovePendingStatus bit and the MovePendingQueueFullStatus bit are always the same.

Master Driven Speed Control (MDSC) and Motion Direct Command Support

The Motion Direct commands are not available in the instruction tree for the MCCM instruction. You must program an MCCM in one of the supported programming languages before you execute an MAM or MAJ in Time Driven Mode. A runtime error will occur if an MCCM is not previously executed in an MAM and MAJ in Master Driven Mode.

Motion Coordinated Change Dynamics (MCCD)

The Motion Coordinated Change Dynamics (MCCD) instruction starts a change in the path dynamics of the specified coordinate system. Based upon the Motion Type, the MCCD changes the coordinated motion profile that is currently active on the system.

ATTENTION: Use each tag for the motion control attribute of instructions only once. Re-use of the motion control tag in other instructions can cause unintended operation. This can result in damage to equipment or personal injury.

ATTENTION: Risk of Velocity and/or End Position Overshoot

If you change move parameters dynamically by any method, that is by changing move dynamics (MCD or MCCD) or by starting a new instruction before the last one has completed, be aware of the risk of velocity and/or end position overshoot.

A Trapezoidal velocity profile can overshoot if maximum deceleration is decreased while the move is decelerating or is close to the deceleration point.

An S-Curve velocity profile can overshoot if either:

- maximum deceleration is decreased while the move is decelerating or close to the deceleration point.
- maximum acceleration jerk is decreased and the axis is accelerating. Keep in mind, however, that jerk can be changed indirectly if it is specified in % of time.

Operands

The MCCD instruction supports the following operands:

- Relay Ladder
- Structured Text

Relay Ladder

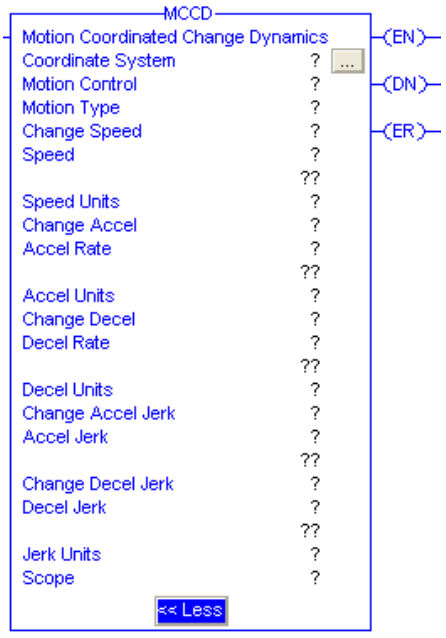


Table 37 - MCCD Instruction Operands - Relay Ladder

Operand	Type	Format	Description
Coordinate System	COORDINATE_SYSTEM	Tag	The Coordinate System operand specifies the set of motion axes that define the dimensions of a coordinate system. The coordinate system supports up to three primary axes.
Motion Control	MOTION_ INSTRUCTION	Tag	Structure used to access instruction status parameters.
Motion Type	SINT, INT, or DINT	Immediate	1 = Coordinated Move
Change Speed	SINT, INT, or DINT	Immediate	The Change Speed operand determines whether or not to change the speed of the coordinated motion profile. 0 = No - no change is made to the speed of the coordinated motion. 1 = Yes - the speed of the coordinated motion is changed by the value defined in the Speed and Speed Units operands.
Speed	SINT, INT, DINT, or REAL	Immediate or tag	The Speed operand defines the maximum speed along the path of the coordinated move. [coordination units]
Speed Units	SINT, INT, or DINT	Immediate	The Speed Units operand defines the units applied to the Speed operand either directly in coordination units of the specified coordinate system or as a percentage of the maximum values defined in the coordinate system. 0 = Units per Sec 1 = % of Maximum 4 = Units per MasterUnit
Change Accel	SINT, INT, or DINT	Immediate	The Change Accel operand determines whether or not to change the acceleration of the coordinated motion profile. 0 = No - no change is made to the acceleration of the coordinated motion. 1 = Yes - the acceleration of the coordinated motion is changed by the value defined in the Accel Rate and Accel Units operands.
Accel Rate	SINT, INT, DINT, or REAL	Immediate or tag	The Accel Rate operand defines the maximum acceleration along the path of the coordinated move. [coordination units]
Accel Units	SINT, INT, or DINT	Immediate	The Accel Units operand defines the units applied to the Accel Rate operand either directly in coordination units of the specified coordinate system or as a percentage of the maximum values defined in the coordinate system. 0 = Units per Sec^2 1 = % of Maximum 4 = Units per $MasterUnit^2$
Change Decel	SINT, INT, or DINT	Immediate	The Change Decel operand determines whether or not to change the deceleration of the coordinated motion profile. 0 = No - no change is made to the deceleration of the coordinated motion. 1 = Yes - the deceleration of the coordinated motion is changed by the value defined in the Decel Rate and Decel Units operands.
Decel Rate	SINT, INT, DINT, or REAL	Immediate or tag	The Decel Rate operand defines the maximum deceleration along the path of the coordinated move. [coordination units]
Decel Units	SINT, INT, or DINT	Immediate	The Decel Units operand defines the units applied to the Decel Rate operand either directly in coordination units of the specified coordinate system or as a percentage of the maximum values defined in the coordinate system. 0 = Units per Sec^2 1 = % of Maximum 4 = Units per $MasterUnit^2$
Change Accel Jerk	SINT, INT, or DINT	Immediate	The Change Accel Jerk operand determines whether or not to change the acceleration jerk of the coordinated motion profile. 0 = No - no change is made to the acceleration jerk of the coordinated motion. 1 = Yes - the acceleration of the coordinated motion is changed by the value defined in the Accel Jerk Rate and Jerk Units operands.

Table 37 - MCCD Instruction Operands - Relay Ladder

Operand	Type	Format	Description
Accel Jerk	SINT, INT, DINT, or REAL	Immediate or tag	Accel Jerk defines the maximum acceleration jerk for the programmed move. For more information on calculating Accel Jerk, see Jerk Units section below. You **must** always enter a value for the Accel Jerk operand. This instruction only uses the value if the Profile is configured as S-Curve. Accel Jerk is the acceleration jerk rate for the coordinate system. Use these values to get started: • Accel Jerk = 100 (% of Time) • Jerk Units = 2
Change Decel Jerk	SINT, INT, or DINT	Immediate	The Change Decel Jerk operand determines whether or not to change the deceleration jerk of the coordinated motion profile. 0 = No - no change is made to the deceleration jerk of the coordinated motion. 1 = Yes - the deceleration of the coordinated motion is changed by the value defined in the Accel Jerk Rate and Jerk Units operands
Decel Jerk	SINT, INT, DINT, or REAL	Immediate or Tag	Decel Jerk defines the maximum deceleration jerk for the programmed move. You **must** always enter a value for the Decel Jerk operand. This instruction only uses the value if the Profile is configured as S-Curve. • Decel Jerk is the deceleration jerk rate for the coordinate system. Use these values to get started. Decel Jerk = 100 (% of Time) • Jerk Units = 2
Jerk Units	SINT, INT, or DINT	Immediate	The jerk units define the units that are applied to the values entered in the Accel Jerk and Decel Jerk operands. 0 = Units per sec^3 1 = % of Maximum 2 = % of Time (use this value to get started) 4 = Units per MasterUnit3 6 = % of Time Master Driven Refer to Convert Jerk Units on page 125.
Scope	SINT, INT, or DINT	Immediate	Choosing Active Motion for the Scope operand specifies that the changes affect only the motion dynamics of the active coordinated motion instruction. Choosing Active and Pending Motion specifies that the changes affect the motion dynamics of the active coordinated motion instruction and any pending coordinated motion instruction in the queue. Currently the queue size is limited to one instruction after the active instruction.

```
MCCD(CoordinateSystem,
MotionControl,MotionType
ChangeSpeed,Speed,SpeedUnits,ChangeAccel,AccelRate,
AccelUnits,ChangeDecel,
DecelRate,DecelUnits,ChangeAccelJerk,AccelJerk,Change
DecelJerk,DecelJerk,JerkUnits,
Scope);
```

Structured Text

The operands are the same as those for the relay ladder MCCD instruction.

When entering enumerations for the operand value in structured text, multiple word enumerations must be entered without spaces. For example, enter Decel Units as unitspersec^2 rather than Units per Sec^2 as displayed in the ladder logic.

For the operands that have enumerated values, enter your selection as follows.

This Operand	**Has These Options That You Enter as**	
	Text	**Or as**
Coordinate System	No enumeration	Tag
Motion Control	No enumeration	Tag
Move Type	No enumeration	Tag 0 = Absolute 1 = Incremental
ChangeSpeed	No Yes	0 1
Speed	No enumeration	Immediate or tag
SpeedUnits	Unitspersec %ofmaximum Unitspermasterunit	0 1 4
ChangeAccel	No Yes	0 1
Accel Rate	No enumeration	Immediate or tag
Accel Units	Unitspersec^2 %ofmaximum unitspermasterunit^2	0 1 4
ChangeDecel	No Yes	0 1
Decel Rate	No enumeration	Immediate or tag
Decel Units	Unitspersec^2 %ofmaximum unitspermasterunit^2	0 1 4
Change Accel Jerk	No enumeration	0 = No 1 = Yes
Accel Jerk	No enumeration	You **must** always enter a value for the Accel Jerk operand. This instruction only uses the value if the Profile is configured as S-Curve. Accel Jerk is the acceleration jerk rate for the coordinate system. Use these values to get started. • Accel Jerk = 100 (% of Time)
Change Decel Jerk	No enumeration	0 = No 1 = Yes

This Operand	Has These Options That You Enter as	
	Text	**Or as**
Decel Jerk	No enumeration	Immediate or tag You **must** always enter a value for the Decel Jerk operand. This instruction only uses the value if the Profile is configured as S-Curve. Use these values to get started. • Decel Jerk = 100 (% of Time) Jerk Units = 2
Jerk Units	Unitspersec^3 %ofmaximum %oftime unitspermasterunit^3 %oftime-masterdriven	0 1 2 (use this value to get started) 3 6
Scope	No enumeration	0 = Active Motion 1 = Active and Pending Motion

Executing the Instruction

MCCD is a transitional instruction.

- In relay ladder, toggle the rung-condition-in from cleared to set each time the instruction executes.
- In structured text, condition the instruction so that it only executes on a transition.

Motion Control Bits

The following control bits are affected by the MCCD instruction.

Mnemonic	Description
.EN (Enable) Bit 31	The Enable bit is set when the rung transitions from false to true. It resets when the rung transitions from true to false.
.DN (Done) Bit 29	The Done bit resets when the rung transitions from false to true. It sets when target position is calculated successfully.
.ER (Error) Bit 28	The Error bit resets when the rung transitions from false to true. It sets when target position fails to calculate successfully.

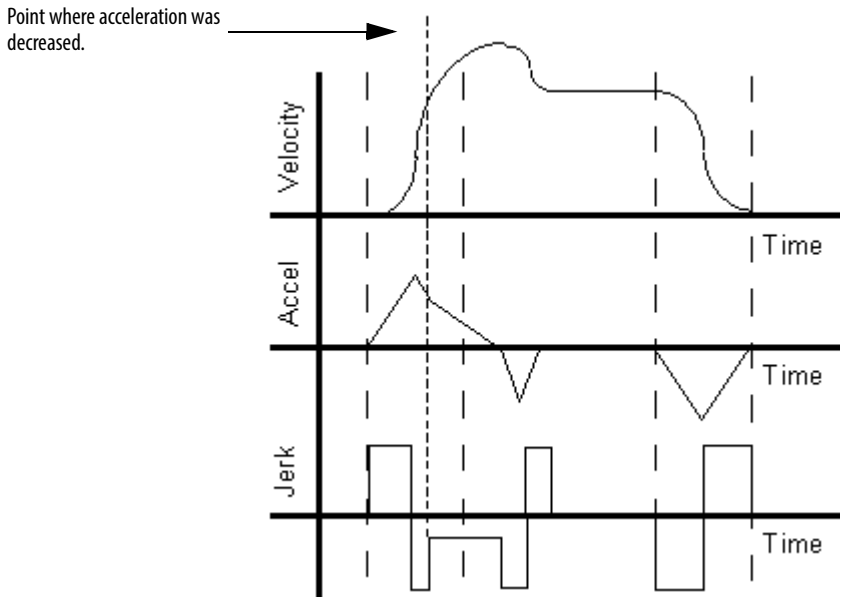
Motion Type

The motion type operand determines which motion profile to change. Coordinated Move is the only available option. When selected, the Coordinated Move option changes the motion of the currently active move in the coordinate system.

Impact of Changes to Acceleration and Deceleration Values on Motion Profile

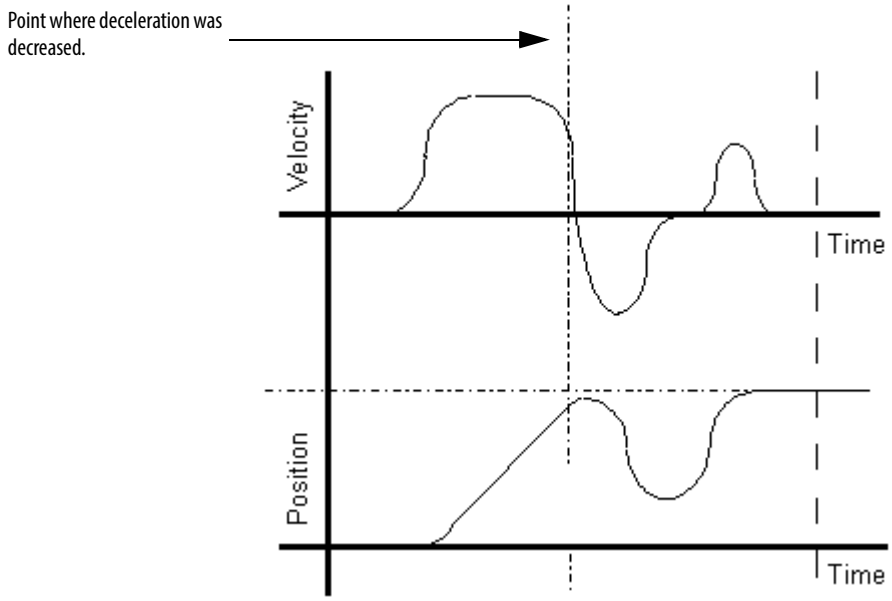
The following graph illustrates what could happen when a MCCD instruction is used to reduce the acceleration as velocity approaches maximum. The new acceleration Jerk Rate becomes smaller, further limiting the maximum change in acceleration. Velocity overshoot occurs due to the additional time required for acceleration to reach zero. Another profile is generated to bring velocity back to the programmed maximum.

Figure 97 - Effect of Change to Acceleration



The Effect of Change to Deceleration graph illustrates what could happen when an MCCD instruction is used to reduce the deceleration as velocity and position approach their target endpoints. The new deceleration Jerk Rate becomes smaller. The time required to decelerate to zero causes velocity to undershoot, passing through zero and becoming negative. Axis motion also reverses direction until velocity returns to zero. An additional profile is generated to bring position back to the programmed target.

Figure 98 - Effect of Change to Deceleration



Arithmetic Status Flags

Not affected.

Fault Conditions

None.

Error Codes

See Error Codes (ERR) for Coordinate Motion Instructions on page 261.

Runtime Error Condition

For the Master Driven Speed Control (MDSC) function, an error occurs at runtime if you attempt to change the mode of the system from Master Driven to Time Driven or from Time Driven to Master Driven.

Extended Error Codes

Extended Error codes help to further define the error message given for this particular instruction. Their behavior is dependent upon the Error Code with which they are associated.

The Extended Error Codes for Servo Off State (5), Shutdown State (7), Axis Type Not Servo (8), Axis Not Configured (11), Homing In Process Error (16), and Illegal Axis Data type (38) errors all function in the same fashion. A number between 0...*n* is displayed for the Extended Error Code. This number is the index to the Coordinate System indicating the axis that is in the error condition.

For the MCCD instruction, Error Code 13 - Parameter Out of Range, Extended Errors return a number that indicates the offending parameter as listed on the faceplate in numerical order from top to bottom beginning with zero. For example, 2 indicates the parameter value for Move Type is in error.

Table 38 - Error Code 13 MCCD Extended Errors

Referenced Error Code and Number	Extended Error Numeric Indicator	Instruction Parameter	Description
Parameter Out Of Range (13)	2	Move Type	Move Type is either less than 0 or greater than 1.
Parameter Out Of Range (13)	4	Speed	Speed is less than 0.
Parameter Out Of Range (13)	7	Accel Rate	Accel Rate is less than or equal to 0.
Parameter Out Of Range (13)	10	Decel Rate	Decel Rate is less than or equal to 0.

Error Code 54 – Maximum Deceleration Value is Zero

If the Extended Error returns a positive number (0-*n*) it's referring to the offending axis in the coordinate system.

1. Go to the Coordinate System Properties General Tab and look under the Brackets ([])column of the Axis Grid to determine which axis has a Maximum Deceleration value of 0.
2. Click the ellipsis next to the offending axis to access the Axis Properties screen.
3. Go to the Dynamics tab and make the appropriate change to the Maximum Deceleration Value.

   If the Extended Error number is -1, this means the Coordinate System has a Maximum Deceleration Value of 0.
4. Go to the Coordinate System Properties Dynamics Tab to correct the Maximum Deceleration value.

MCCD Changes to Status Bits

For the Master Driven Speed Control (MDSC) function, when the MCCD is executed (goes IP), the CalculatedDataAvailable (CDA) status bit is cleared (as specified by the Scope variable of the MCCD instruction) in each MCLM and MCCM instruction tag, which indicates that the Event Distances has been computed. (The Scope variable specifies either the Active Motion instruction or Active Motion and Pending instruction (that is, all instructions, in the queue)).

After the MCCD is complete and the Event Distances have been recomputed, the CalculatedDataAvailable status bit is set again. Therefore, look at the CalculatedDataAvailable status bit after the MCCD instruction has been completed to determine when to use the recomputed Event Distances.

If a MCCD is executed (goes IP), the CDA bit is cleared. The Calculated Data for the move is recomputed by using the new dynamics parameters. The CDA bit is set again when computations are complete. The Calculated Data that is recomputed is measured from the original Motion Start Point (MSP) to the Event Distance point by using the new dynamics parameters as changed by the MCCD instruction - not from the point of the MCCD.

Note that if the MCCD changes the speed to 0, the Event Distance is not recomputed; the CDA bit is not set. The Event Distance is however recomputed if a second MCCD is issued to restart the motion. The recomputed Calculated Data includes the duration of the stopped motion.

If the Event Distance is set to 0, the Calculated Data is set to equal the position that equals the length of the move. This can be one or two coarse update periods before the PC bit is set because of an internal delay. The end position is typically achieved in the middle of a coarse update period, which adds up to one additional coarse update period to the delay. Therefore, if the master is moved a distance equal to the Calculated Data, you must wait up to 2 iterations more for the PC bit of the slave move to be set.

Motion Coordinated Stop (MCS)

The Motion Coordinated Stop (MCS) instruction initiates a controlled stop of coordinated motion. Any pending motion profiles are cancelled.

ATTENTION: Use each tag for the motion control attribute of instructions only once. Re-use of the motion control tag in other instructions can cause unintended operation. This can result in damage to equipment or personal injury.

ATTENTION: Risk of Velocity and/or End Position Overshoot

If you change move parameters dynamically by any method, that is by changing move dynamics (MCD or MCCD) or by starting a new instruction before the last one has completed, be aware of the risk of velocity and/or end position overshoot.

A Trapezoidal velocity profile can overshoot if maximum deceleration is decreased while the move is decelerating or is close to the deceleration point.

An S-Curve velocity profile can overshoot if either:

- maximum deceleration is decreased while the move is decelerating or close to the deceleration point.
- maximum acceleration jerk is decreased and the axis is accelerating. Keep in mind, however, that jerk can be changed indirectly if it is specified in % of time.

Operands

The MCS instruction supports the following operands:

- Relay Ladder

- Structured Text

Relay Ladder

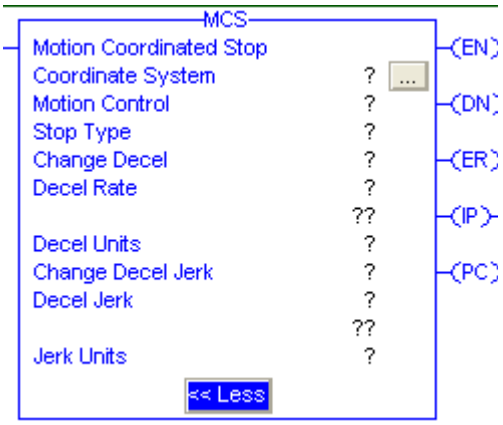


Table 39 - MCS Instruction Operands - Relay Ladder

Operand	Type	Format	Description	
Coordinate System	COORDINATE_SYSTEM	Tag	The Coordinate System operand specifies the set of motion axes that define the dimensions of a coordinate system. The coordinate system supports up to three primary axes.	
Motion control	MOTION_ INSTRUCTION	Tag	Control tag for the instruction	
Stop Type	DINT	Immediate	**If you want to**	**Choose this Stop Type**
			Stop all motion for the axes of the coordinate system and stop any transform that the coordinate system is a part of	All (0) - For each axis, all motion generators, including the coordinated motion, are taken into account when computing the initial dynamics (for example, acceleration rate and velocity) to be used in the Decel. Every axis in the coordinated system is stopped independently by using the computed initial dynamics.
			Stop only coordinated moves	Coordinated Move (2)
			Cancel any transform that the coordinate system is a part of	Coordinated Transform (3)
Change Decel[1]	DINT	Immediate	**If you want to**	**Then choose**
			Use the maximum deceleration rate of the coordinate system	No (0)
			Specify the deceleration rate	Yes (1)
Decel Rate	REAL	Immediate or tag	**Important**: An axis could overshoot its target position if you reduce the deceleration while a move is in process. Deceleration along the path of the coordinated move. The instruction uses this value: • Only if Change Decel is Yes. • Only for coordinated moves. Enter a value greater than 0.	
Decel Units	DINT	Immediate	0 = Units per sec^2 1 = % of Maximum	

Table 39 - MCS Instruction Operands - Relay Ladder

Operand	Type	Format	Description
Change Decel Jerk	SINT, INT, or DINT	Immediate	0 = No 1 = Yes
Decel Jerk	SINT, INT, DINT, or REAL	Immediate or tag	You **must** always enter a value for the Decel Jerk operand. This instruction only uses the value if the Profile is configured as S-Curve. Decel Jerk is the deceleration jerk rate for the coordinate system. Use these values to get started. • Decel Jerk = 100 (% of Time) • Jerk Units = 2
Jerk Units	SINT, INT, or DINT	Immediate	0 = Units per sec^3 1 = % of Maximum 2 = % of Time (use this value to get started)

(1) Overshoot can occur if MCS is executed close to or beyond the deceleration point and the deceleration limit is decreased. Keep in mind, that deceleration can be decreased indirectly by setting ChangeDecel to NO if configured maximum deceleration rate is less than that the active deceleration rate."

```
MCS(CoordinateSystem,
MotionControl,StopType, ChangeDecel,
DecelRate,DecelUnits,
ChangeDecelJerk,DecelJerk,
JerkUnits);
```

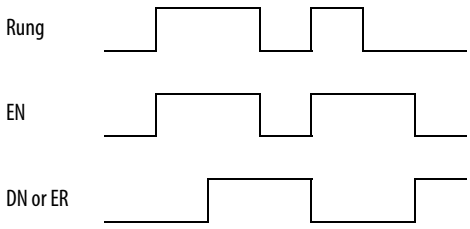
Structured Text

The structured text operands are the same as the ladder diagram operands. Enter the stop type and decel units without spaces.

Enter the Coordinate System operand as CoordinateSystem.

Motion Control Bits

Table 40 - MCS Bit Types and Functions

To see if	Check if this bit is on	Data type	Notes
The rung is true.	EN	BOOL	Sometimes the EN bit stays on even if the rung goes false. This happens if the rung goes false before the instruction is complete or has encountered an error. 
The stop was successfully initiated.	DN	BOOL	
An error happened.	ER	BOOL	
The axis is stopping.	IP	BOOL	Any of these actions ends the MCS instruction and turns off the IP bit: • The coordinate system is stopped. • Another MCS instruction supersedes this MCS instruction. • Shutdown instruction. • Fault Action.
The axis is stopped.	PC	BOOL	The PC bit stays on until the rung makes a false-to-true transition.

How Stop Types Affect Transforms

The following table describes how the stop types affect coordinate systems that are a part of a transform.

Table 41 - MCS Stop Types

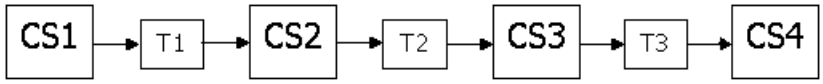
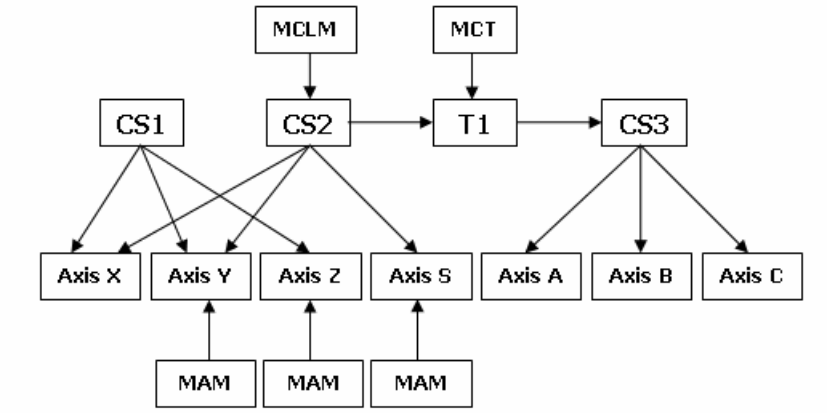
Stop types	Description
All	This stop type: • stops the axes in the specified coordinate system. It also stops the axes of any coordinate system that shares axes with this coordinate system. • cancels any transforms that the coordinate system is a part of.
Coordinated Move	This stop type stops only the coordinated moves. Any transforms stay active.
Coordinated Transform	This stop type cancels the transforms associated with the specified coordinate system. All transform-related motion stops on all associated target coordinate systems. However, source coordinate axes continue to move as instructed. **Example** If four coordinate systems are linked via three transforms, and the first coordinate system (CS1) is the source and is processing commanded motion.  Executing an MCS instruction on CS2 and using a stop type of coordinated transform results in: • Transforms T1 and T2 are canceled. • Transform T3 stays active. • the axes in CS1 stay in motion. • the axes in Coordinate Systems CS2 and CS3 stop via the deceleration rate selected in the MCS instruction or the maximum coordinate deceleration rate. • the axes in CS4 follow the respective CS3 axes. In a Motion Axis Stop (MAS) instruction, a stop type of all also cancels transforms.

Figure 99 - How Stop Types Affect Transforms and Axis Motion Example

Suppose you have this situation.



Where:

- coordinate system 1 (CS1) contains the X, Y, and Z axes.
- coordinate system 2 (CS2) contains the Y, Z, and S axes.
- coordinate system 3 (CS3) contains the A, B, and C axes.
- transform (T1) links source coordinate CS2 to target CS3.

- CS2 (XYS) axes are mapped to CS3 (ABC) axes.
- MAM instructions executed on the Y, Z, and S axes.
- MCLM instruction executed on CS2.
- MCT instruction executed with CS2 as the source and CS3 as the target.
- No coordinate instructions were executed on CS2 or CS3.

Table 42 - Results of Stop Type and Instruction Interaction

Instruction	Stop Type	Result
MCS on CS1	All	The MCLM instruction on CS2 stops.
		The MAM on Y stops.
		The MAM on Z stops.
		The MAM on S continues.
		T1 is canceled.
		Axes ABC stops due to canceling the transform.
MCS on CS2	All	The MCLM instruction on CS2 stops.
		The MAM on Y stops.
		The MAM on S stops.
		The MAM on Z continues.
		T1 is canceled.
		Axes ABC stops due to canceling the transform.
MCS on CS3	All	The MCLM instruction on CS2 continues.
		The MAM on Y continue.
		The MAM on S continues
		The MAM on Z continues.
		T1 is canceled.
		Axes ABC stop due to canceling the transform.
MCS on CS1	Coordinated Move	The MCLM instruction on CS2 continues.
		The MAM on Y continues.
		The MAM on S continues.
		The MAM on Z continues.
		T1 stays active.
		Axes ABC follow the respective CS2 axes.
MCS on CS2	Coordinated Move	The MCLM instruction on CS2 stops.
		The MAM on Y continues.
		The MAM on S continues.
		The MAM on Z continues.
		T1 stays active.
		Axes ABC follow the respective CS2 axes.

Table 42 - Results of Stop Type and Instruction Interaction

Instruction	Stop Type	Result
MCS on CS3	Coordinated Move	The MCLM instruction on CS2 continues.
		The MAM on Y continues.
		The MAM on S continues.
		The MAM on Z continues.
		T1 stays active.
		Axes ABC follow the respective CS2 axes.
MAS on Y	All	The MCLM instruction on CS2 stops.
		The MAM on Y stops.
		The MAM on S continues.
		The MAM on Z continues.
		T1 is canceled.
		Axes ABC stop due to canceling the transform.
MAS on Y	Move	The MCLM instruction on CS2 continues.
		The MAM on Y stops.
		The MAM on S continues.
		The MAM on Z continues.
		T1 stays active.
		Axes ABC follow the respective CS2 axes.
MAS on Z	All	The MCLM instruction on CS2 continues.
		The MAM on Y continues.
		The MAM on S continues.
		The MAM on Z stops.
		T1 stays active.
		Axes ABC follow the respective CS2 axes.
MAS on Z	Move	The MCLM instruction on CS2 continues.
		The MAM on Y continues.
		The MAM on S continues.
		The MAM on Z stops.
		T1 stays active.
		Axes ABC follow the respective CS2 axes.
MCS on CS1	Coordinated Transform	MCLM instruction on CS2 continues.
		The MAM on Y continues.
		The MAM on S continues.
		The MAM on Z continues.
		T1 stays active.
		Axes ABC follow the respective CS2 axes.

Table 42 - Results of Stop Type and Instruction Interaction

Instruction	Stop Type	Result
MCS on CS2	Coordinated Transform	T1 is canceled.
		MCLM instruction on CS2 continues.
		The MAM on Y continues.
		The MAM on S continues.
		The MAM on Z continues.
		Axes ABC stop due to canceling the transform.
MCS on CS3	Coordinated Transform	T1 is canceled.
		MCLM instruction on CS2 continues.
		The MAM on Y continues.
		The MAM on S continues.
		The MAM on Z continues.
		Axes ABC stop due to canceling the transform.

Master Driven Speed Control (MDSC) and the MCS Instruction

If an MCS is issued when in Master Driven Mode, a switch is made to Time Driven Mode and the axes are stopped in Time Driven Mode.

MCS All resets the IP bit of the Master Driven Coordinate Control (MDCC) instruction. Other stop types do not reset the IP bit.

The MCS All instruction clears the pending Master Axis for all future coordinated motion instructions. However, MCS ALL on the Master axis does not break the MDSC link.

The AC bit of the MDCC instruction is reset when the axis is stopped.

The instruction queue is cleared when an MCS All or MCS Coordinated is executed (goes IP).

The status bit CalculatedDataAvailable in an active motion instruction status word for an MCLM or MCCM instruction clears when an MCS is executed (goes IP). The CalculatedData is not recomputed.

Note that if a stop is issued very close to the programmed endpoint, the actual stop can be beyond the programmed endpoint, especially if run in Master Driven Mode.

Arithmetic Status Flags

Not affected.

Fault Conditions

None.

Error Codes

See Error Codes (ERR) for Coordinate Motion Instructions on page 261.

Extended Error Codes

See Error Codes (ERR) for Coordinate Motion Instructions on page 261. It has information about how to use the extended error codes.

MCS Changes to Status Bits

The instruction changes these status bits when it executes.

Table 43 - MCS Status Bit Tags and Types

In the tag for the	This bit	When the stop type is	Turns
Axis	CoordinatedMotionStatus	→	Off when the coordinated move stops
	TransformStateStatus	Coordinated Move	Unchanged
		• All • Coordinated Transform	Off
	ControlledByTransformStatus	Coordinated Move	Off when the axes stop and the PC bit of the MCS instruction turns on
		• All • Coordinated Transform	Off
Coordinate system	MotionStatus	→	Off when the coordinated move stops
	AccelStatus	→	Off
	DecelStatus	→	On during the stop and then off when the stop completes
	StoppingStatus	→	On during the stop and then off when the PC bit turns on
	MoveStatus	→	Off
	MoveTransitionStatus	→	Off
	MovePendingStatus	→	Off
	TransformSourceStatus	Coordinated Move	Unchanged
		• All • Coordinated Transform	Off
	TransformTargetStatus	Coordinated Move	Unchanged
		• All • Coordinated Transform	Off

Table MCS and MACS Instructions with Stop Types shows the results of executing various MCS and MAS instructions with different stop types.

Motion Coordinated Shutdown (MCSD)

Use the Motion Coordinated Shutdown (MCSD) instruction to perform a controlled shutdown of all the axes in the named coordinate system.

ATTENTION: Use each tag for the motion control attribute of instructions only once. Re-use of the motion control tag in other instructions can cause unintended operation. This can result in damage to equipment or personal injury.

ATTENTION: Risk of Velocity and/or End Position Overshoot

If you change move parameters dynamically by any method, that is by changing move dynamics (MCD or MCCD) or by starting a new instruction before the last one has completed, be aware of the risk of velocity and/or end position overshoot.

A Trapezoidal velocity profile can overshoot if maximum deceleration is decreased while the move is decelerating or is close to the deceleration point.

An S-Curve velocity profile can overshoot if either:

- maximum deceleration is decreased while the move is decelerating or close to the deceleration point.
- maximum acceleration jerk is decreased and the axis is accelerating. Keep in mind, however, that jerk can be changed indirectly if it is specified in % of time.

Operands

The MCSD instruction supports the following operands:

- Relay Ladder
- Structured Text

Relay Ladder

Table 44 - Operands Relay Ladder

Operand	Type	Format	Description
Coordinate System	COORDINATE_SYSTEM	Tag	The Coordinate System operand specifies the set of motion axes that define the dimensions of a coordinate system. The coordinate system supports up to three primary axes. Only the axes configured as primary axes (up to 3) are included in the coordinate velocity calculations.
Motion Control	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.

MCSD(CoordinateSystem, MotionControl);

Structured Text

The operands are the same as those for the relay ladder MCSD instruction.

Executing the Instruction

MCSD is a transitional instruction.

- In relay ladder, toggle the rung-condition-in from cleared to set each time the instruction executes.
- In structured text, condition the instruction so that it only executes on a transition.

Motion Control Bits

The following control bits are affected by the MCSD instruction.

Table 45 - MCSD Motion Control Mnemonics

Mnemonic	Description
.EN (Enable) Bit 31	The Enable bit sets when the rung transitions from false to true. It resets when the rung goes from true to false.
.DN (Done) Bit 29	The Done bit sets when the coordinated shutdown is successfully initiated. It resets when the rung transitions from false to true.
.ER (Error) Bit 28	The Error bit sets when the coordinated shutdown fails to initiate successfully. It resets when the rung transitions from false to true.

Master Driven Speed Control (MDSC) and the MCSD Instruction

When the coordinate system is shut down:

- The IP bit of the Master Driven Coordinate Control (MDCC) instruction is reset on an axis that is shutdown.
- The AC bit of the MDCC instruction resets when the axis is stopped as it is shutdown.
- The MCSD instruction clears the pending Master Axis for all future coordinate system motion instructions.

Arithmetic Status Flags

Not affected.

Fault Conditions

None.

Error Codes

See Error Codes (ERR) for Coordinate Motion Instructions on page 261.

MCSD Changes to Status Bits

Status bits provide a means for monitoring the progress of the motion instruction. There are three types of Status bits that provide pertinent information. They are Axis Status bits, Coordinate System Status bits, and Coordinate Motion Status bits. When the MCS instruction initiates, the status bits undergo the following changes.

Table 46 - Axis Status Bits

Bit Name	Effect
CoordinatedMoveStatus	Cleared

Table 47 - Coordinate System Status Bits

Bit Name	Effect
ShutdownStatus	Sets when MCSD is executed and all associated axes are shutdown.
ReadyStatus	Cleared after MCSD executes.

Table 48 - Coordinate Motion Status Bits

Bit Name	Effect
AccelStatus	Cleared after MCSD executes.
DecelStatus	Cleared after MCSD executes.
ActualPosToleranceStatus	Cleared after MCSD executes.
CommandPosToleranceStatus	Cleared after MCSD executes.
StoppingStatus	Cleared after MCSD executes.
MoveStatus	Cleared after MCSD executes.
MoveTransitionStatus	Cleared after MCSD executes.
MovePendingStatus	Cleared after MCSD executes.
MovePendingQueueFullStatus	Cleared after MCSD executes.

Motion Coordinated Transform (MCT)

Use the MCT instruction to start a transform that links two coordinate systems together. This is like bi-directional gearing. One way to use the transform is to move a non-Cartesian robot to Cartesian positions.

ATTENTION: Use each tag for the motion control attribute of instructions only once. Re-use of the motion control tag in other instructions can cause unintended operation. This can result in damage to equipment or personal injury.

IMPORTANT You can use this instruction with the following controllers:

1756-L6x controllers

1756-L7x controllers

1769-L18ERM controller

1769-L27ERM controller

1769-L30ERM controller

1769-L33ERM controller

1769-L36ERM controller.

Operands

The MCT instruction supports the following operands:

- Relay Ladder
- Structured Text

Relay Ladder

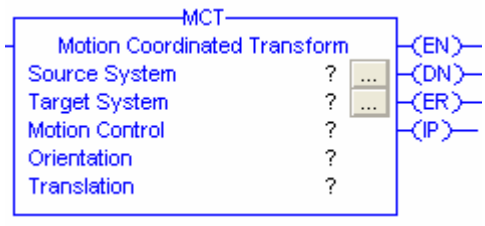


Table 49 - MCT Operand Descriptions — Relay Ladder

Operand	Type	Format	Description	
Source System	COORDINATE_SYSTEM	Tag	Coordinate system that you use to program the moves. Typically, this is the Cartesian coordinate system.	
Target System	COORDINATE_SYSTEM	Tag	Non-Cartesian coordinate system that controls the actual equipment	
Motion Control	MOTION_INSTRUCTION	Tag	Control tag for the instruction	
Orientation	REAL[3]	Array	Do you want to rotate the target position around the X1, X2, or X3 axis?	
			If	**Then**
			No	Leave the array values at zero.
			Yes	Enter the degrees of rotation into the array. Put the degrees of rotation around X1 in the first element of the array, and so on.
			Use an array of three REALs even if a coordinate system has only one or two axes.	
Translation	REAL[3]	Array	Do you want to offset the target position along the X1, X2, or X3 axis?	
			If	**Then**
			No	Leave the array values at zero.
			Yes	Enter the offset distances into the array. Enter the offset distances in coordination units. Put the offset distance for X1 in the first element of the array, and so on.
			Use an array of three REALs even if a coordinate system has only one or two axes.	

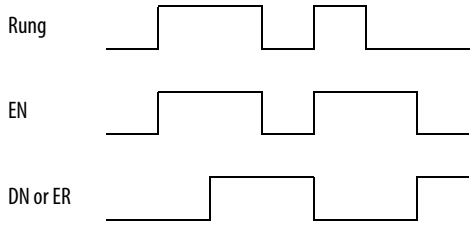
```
MCT(Source System, Target System, Motion Control,
Orientation, Translation);
```

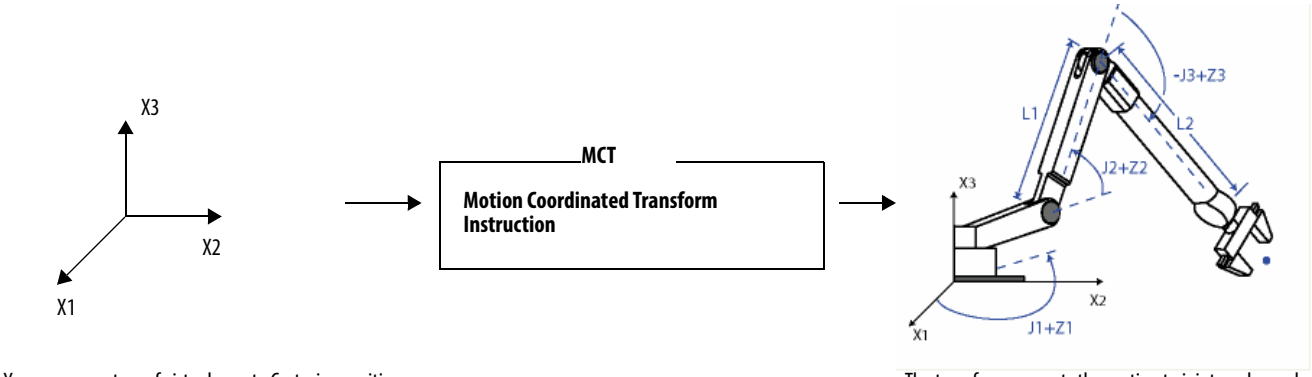
Structured Text

The structured text operands are the same as the ladder diagram operands.

Motion Control Bits

Table 50 - MCT Bit Types and Descriptions

To see if	Check if this bit is on	Data type	Notes
The rung is true.	EN	BOOL	Sometimes the EN bit stays on even if the rung goes false. This happens if the rung goes false before the instruction is done or an error has occurred. 
The instruction is done.	DN	BOOL	The transform process keeps running after the instruction is done.
An error happened.	ER	BOOL	Identify the error number listed in the error code field of the Motion control tag then, see Error Codes (ERR) for Coordinate Motion Instructions on page 261.
The transform process is running.	IP	BOOL	Any of these actions cancels the transform and turns off the IP bit: • Applicable stop instruction • Shutdown instruction • Fault action



You move a system of virtual axes to Cartesian positions (X1, X2, X3).

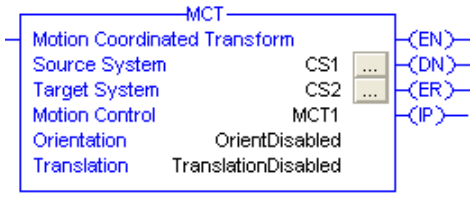
The transform converts the motion to joint angles and moves the robot.

The transform controls up to three joints of the robot: J1, J2, and J3.

Data Flow of MCT Instruction Between Two Coordinate Systems

The following illustrations show the flow of data when an MCT Instruction is active. CS1 is a Cartesian coordinate system containing X1, X2 and X3 axes as the source of the MCT instruction. CS2 is the joint coordinate system containing J1, J2 and J3 axes as the target of the MCT instruction

Figure 100 - Data Flow When a Move is Executed with an MCT Instruction - Forward Transform



Input Data — All axes units are Coordination Units.

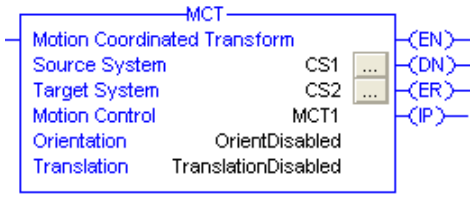
CS2: DATA	SOURCE
Joint Positions (J1, J2, J3)	Machine Real Coordinate System
Link Lengths (L1, L2)	Coordinate System dialog
Base Offsets (X1b, X2b, X3b)	Coordinate System dialog
End Effector Offsets (X1e, X2e, X3e)	Coordinate System dialog
Zero Angle Orientations (Z1, Z2, Z3)	Coordinate System dialog
Orientations (Array [3])	Instruction Faceplate
Translations (Array [3])	Instruction Faceplate

Active Instruction → MCT

Computed Output

CS1: Data	Destination
Cartesian Positions (X1, X2, X3)	Machine Virtual Coordinate System

Figure 101 - Data Flow When a Move is Executed with an MCT Instruction - Inverse Transform



Input Data — All axes units are Coordination Units.

CS1: DATA	SOURCE
Cartesian Positions (X1, X2, X3)	Machine Virtual Coordinate System
Link Lengths (L1, L2)	Coordinate System dialog
Base Offsets (X1b, X2b, X3b)	Coordinate System dialog
End Effector Offsets (X1e, X2e, X3e)	Coordinate System dialog
Zero Angle Orientations (Z1, Z2, Z3)	Coordinate System dialog
Orientations (Array [3])	Instruction Faceplate
Translations (Array [3])	Instruction Faceplate

Active Instruction → MCT

Computed Output

CS2: Data	Destination
Joint Positions (J1, J2, J3)	Machine Real Coordinate System

Programming Guidelines

Follow these guidelines to use an MCT instruction.

ATTENTION: Don't let the robot get fully stretched or fold back on itself. Otherwise it can start to move at a very high speed. In those positions, it loses its configuration as a left or right arm. When that happens, it can start to move at a very high speed.

ATTENTION: Determine the working limits of the robot and keep it within those limits.

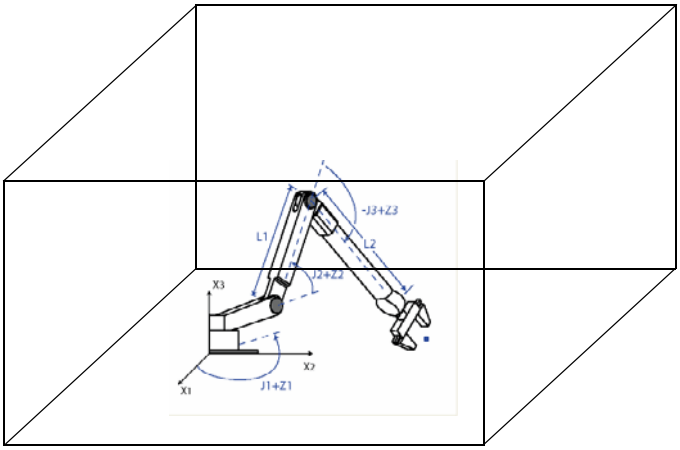
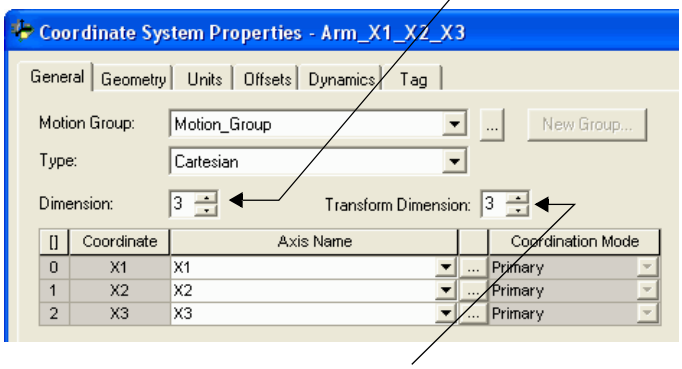
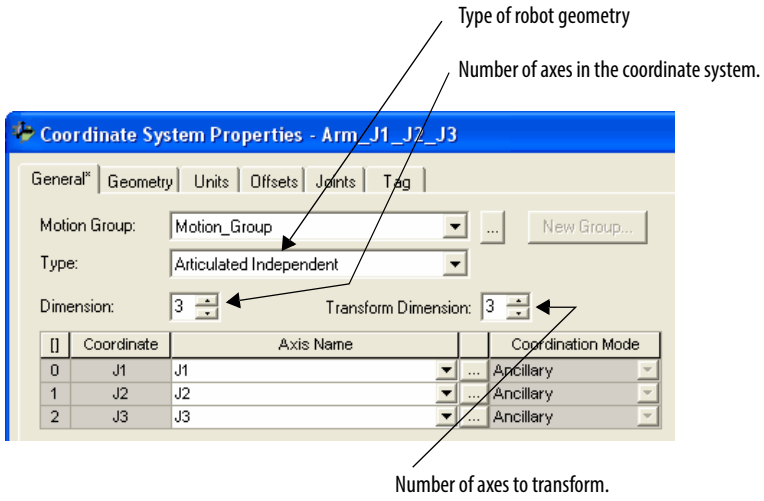
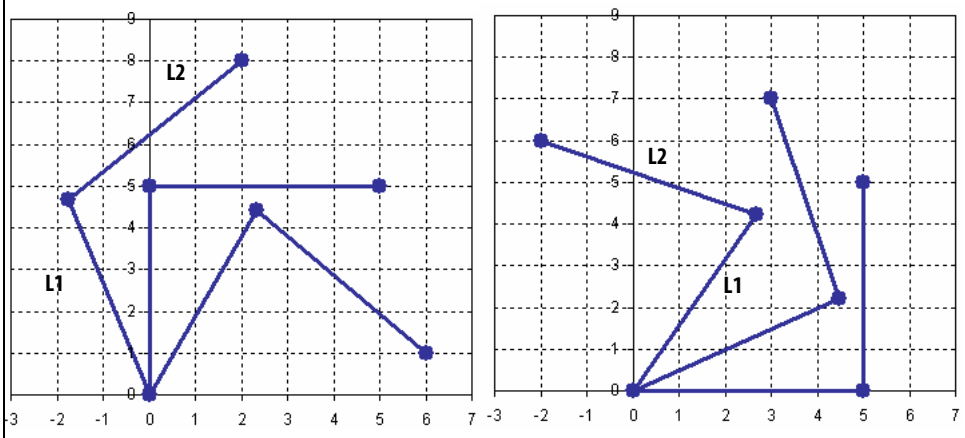
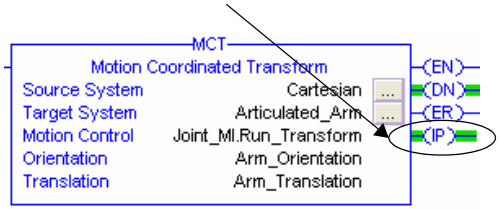
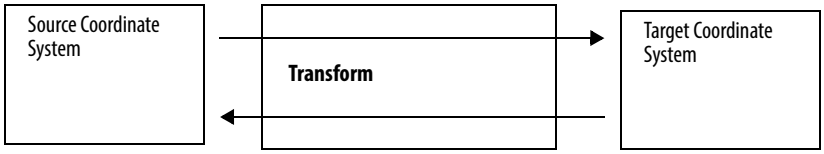
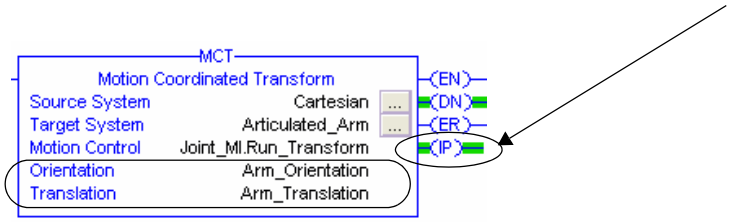


Figure 102 - MCT Programming Guideline Examples

Guideline	Examples and Notes
Set up a coordinate system of axes for the Cartesian positions of the robot. These axes are typically virtual.	**Important**: You can see truncation error in the precision of computations. This happens when both of these conditions are true: Number of axes in the coordinate system.  Number of axes to transform. • The conversion constants of the virtual Cartesian axes in a transformation are small, such as 8000 counts/position unit. • The link lengths of the non-Cartesian coordinate system are small, such as 0.5 inches. It's best to give large conversion constants to the virtual Cartesian axes in a transform, such as 100,000 or 1,000,000 counts/position unit. The maximum travel limit of the robot is $\frac{\pm 2^{31}}{\text{Conversion Constant}}$ Coordination Units

Guideline	Examples and Notes
Set up another coordinate system for the actual joints of the robot.	
Move the robot to a left- or right-arm starting position.	Do you want the robot to move like a left arm or a right arm?  **Left arms** **Right arms** Before you start the transform, move the robot to a resting position that gives it the arm side that you want (left or right). Once you start the transform and initiate a Cartesian move in the Source coordinate system, the robot stays as a left arm or a right arm. If it starts as a left arm, it moves as a left arm. If it starts as a right arm, it moves as a right arm. You can always flip it from a left arm to a right arm or vice versa. To do that, move the joints directly.
Toggle the rung from false to true to execute the instruction.	This is a transitional instruction. In a ladder diagram, toggle the rung-condition-in from false to true each time you want to execute the instruction. When you execute the instruction, the transform starts and the IP bit turns on.  You can let the rung go false once you execute the instruction. The transform stays active.

Guideline	Examples and Notes
In structured text, condition the instruction so that it only executes on a transition. Start the transform before you start any motion.	In structured text, instructions execute each time they are scanned. Condition the instruction so that it only executes on a transition. Use either of these methods: • Qualifier of an SFC action • Structured text construct You cannot start a transform if any motion process is controlling an axis of the source or target coordinate systems. If the MCT instruction is executing and the source side is moving, you cannot also execute a motion instruction to move the target axis or Error 63, Axis In Transform Motion, occurs. The same is true if the target side is moving and you attempt to execute a motion instruction on source axis. With FW revision 24.012 and later, when executing an MCLM Merge All instruction, users can now blend smoothly from a continuous path-linear move to a point-to-point move or vice versa without waiting for the first move to complete, avoiding Error 63. **Example:** Start the transform before you start gearing or camming.
Expect bi-directional motion between the source and target coordinate systems. Use an MCS instruction to cancel the transform.	A transform is bi-directional.  When you start the transform, the position of the source coordinate system changes to match the corresponding position of the target coordinate system. After that, if you move either system, the other system moves in response. The controller continues to control the axes even if you stop scanning the MCT instruction or its rung goes false. Use a Motion Coordinated Stop (MCS) instruction to stop the motion in the coordinate system, cancel the transform, or both.
Execute the MCT instruction again if you change the orientation or translation.	If you want to change orientation or translation values after the transform is running.  Then, execute the instruction again. To execute the instruction, toggle the rung-condition-in from false to true. Also execute the instruction again if you change the geometry of the equipment.

Arithmetic Status Flags

Not affected.

Fault Conditions

None.

Error Codes

See Error Codes (ERR) for Coordinate Motion Instructions on page 261.

Extended Error Codes

Extended Error codes help to further define the error message given for this particular instruction. Their behavior is dependent upon the Error Code with which they are associated.

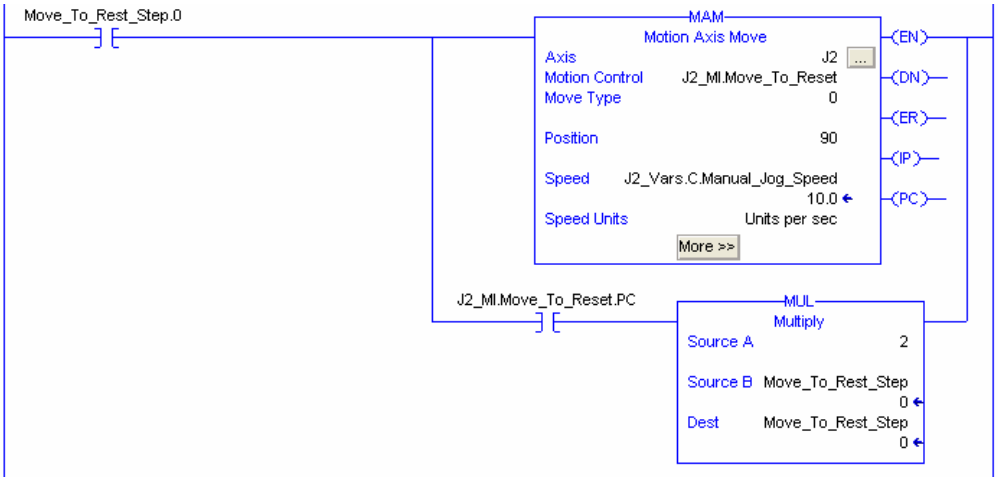
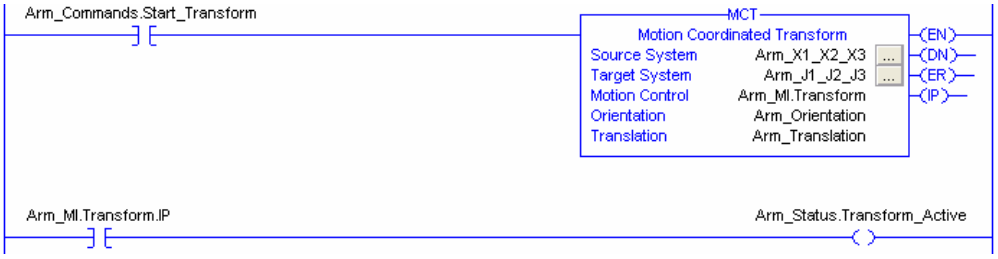
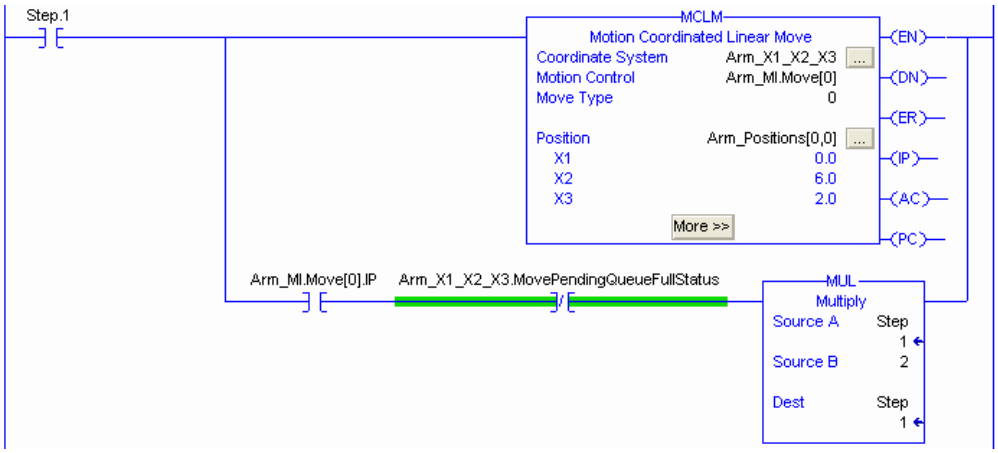
ERR	EXERR	Corrective Action	Notes
61	1	Assign both coordinate systems to the motion group.	
	2	Check that you are using the correct source and target systems.	You cannot use the same coordinate system as source and target.
	3	Set the transform dimension of the source system to the number of axes in the system, up to three.	
	4	Set the transform dimension of the target system to the number of axes in the system to be transformed, up to three.	
	5	Use a different source system.	You can only use one coordinate system as the source for one active transform.
	6	Use a different target system.	You can only use one coordinate system as the target for one active transform.
	7	Look for source or target axes that you are already using in another transform. Use different axes in the coordinate system.	You can only use an axis in one source system and one target system.
	8	Use a target system that isn't the source for this chain of transforms.	You cannot create a circular chain of transforms that leads back to the original source.
	9	Check that you've assigned the correct axes to each coordinate system.	You cannot use the same axes in the source and target systems.
	10	Stop all motion processes for all the axes in both systems (for example, jog, move, and gear).	You cannot start the transform if any motion process is controlling a source or target axis.
	11	Insufficient resources available to initiate the transform connection.	
	12	Set the link lengths.	You cannot use a link length of zero.
	13	Look for source or target axes that are in the shutdown state. Use a Motion Axis Shutdown Reset (MASR) instruction or direct command to reset the axes.	
	14	Uninhibit all the source or target axes.	
	15	Check the configured values for the base offsets and end effector offsets for the Delta or SCARA Delta robot.	(X1b-X1e) can not be less than 0.0 for both the Delta and SCARA Delta robots. For Delta robots, this error can also occur if the value of L1 + (X1b-X1e) is greater than L2.
	16	Check the SCARA independent and SCARA Delta robot configurations to be sure that: • the transform dimension for the source coordinate system is configured as 2. • the configured third axes for the source coordinate system and the target coordinate system are the same.	

MCT Changes to Status Bits

Table 51 - MCT Tag and Bit Functions

To see if	Check the tag for the	And this bit	For
A coordinate system is the source of an active transform.	Coordinate system	TransformSourceStatus	On
A coordinate system is the target of an active transform.	Coordinate system	TransformTargetStatus	On
An axis is part of an active transform.	Axis	TransformStateStatus	On
An axis is moving because of a transform.	Axis	ControlledByTransformStatus	On

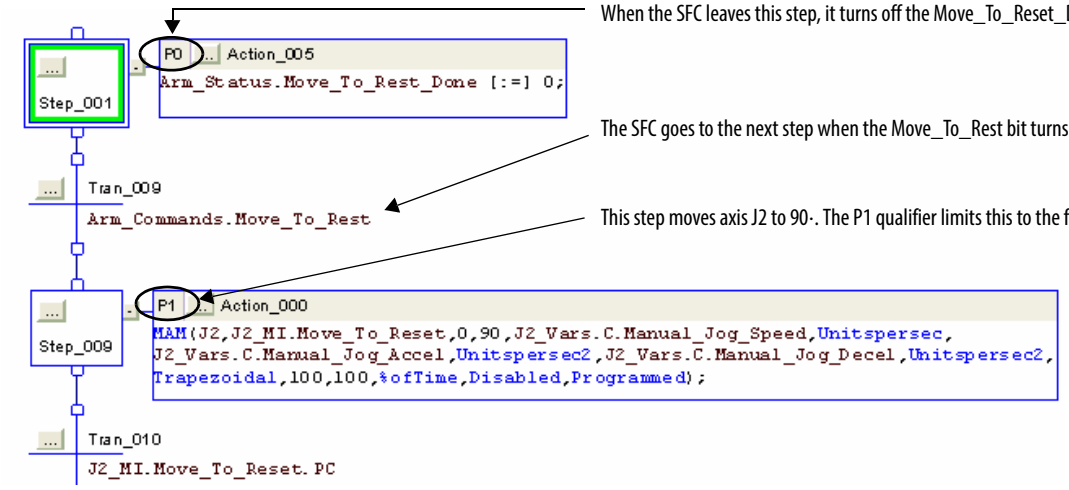
Example 1 - Pick and Place Ladder Diagram

1. Move to rest routine	This routine is a sequence of moves that put an articulated independent robot in an at-rest position at the desired left or right arm angles. When Move_To_Rest_Step.0 turns on, axis J2 moves to 90·. Then the sequence goes to the next step. 
2. Start transform routine	When Arm_Commands.Start_Transform turns on, the transform starts. The IP bit signals that the transform is running. 
3. Pick and place routine	This routine is one in a sequence of MCLM instructions that move the Cartesian system. The joints of the robot follow the moves. When Step.1 turns on, the coordinate system moves to 0, 6, 2. When the move is in process (IP), the sequence queues the next move. 

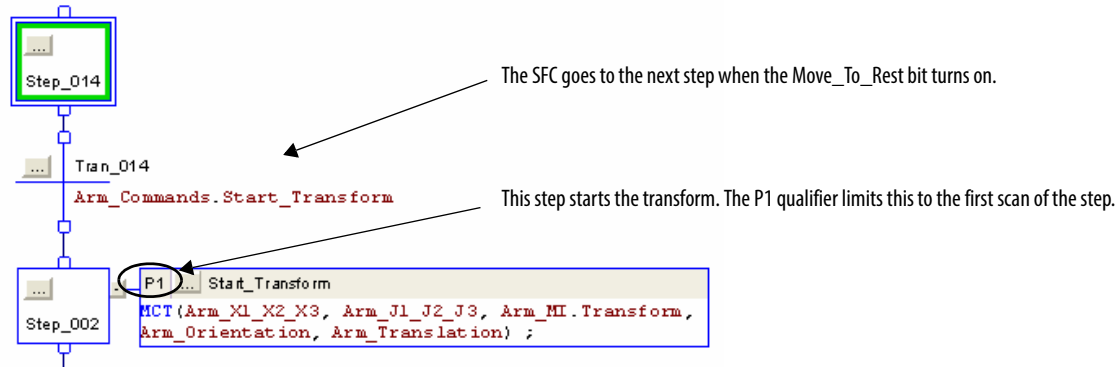
Example 2 - Pick and Place - Structured Text

1. Move to rest routine

This routine is a sequence of moves that put the robot in an at-rest position at the desired left or right arm angles.

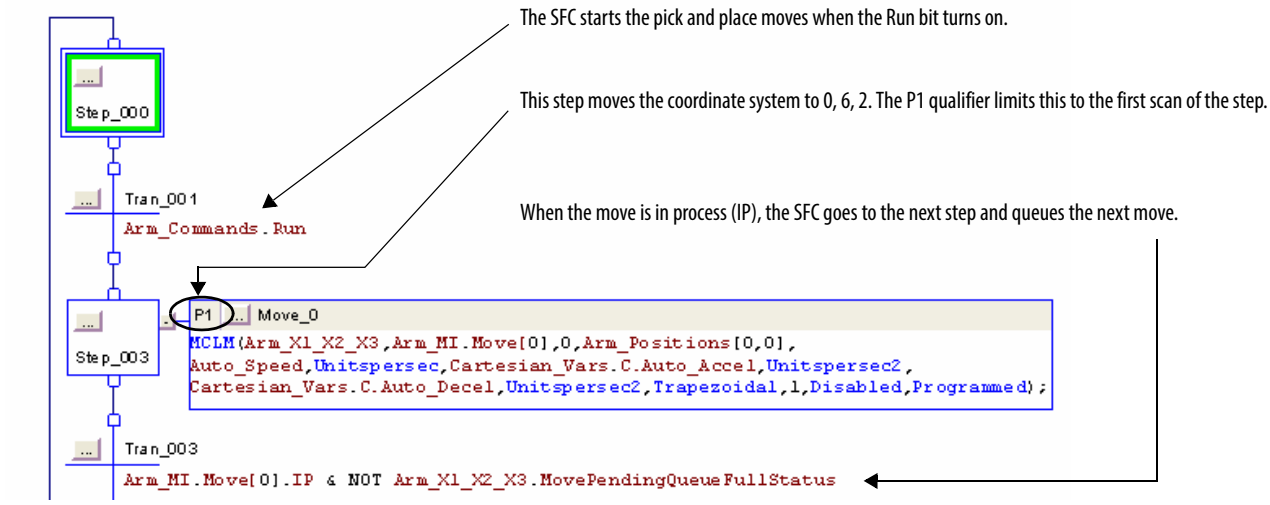


2. Start transform routine


Step_014
The SFC goes to the next step when the Move_To_Rest bit turns on.
Tran_014
Arm_Commands.Start_Transform
This step starts the transform. The P1 qualifier limits this to the first scan of the step.
Step_002
P1
Start_Transform
MCT(Arm_X1_X2_X3, Arm_J1_J2_J3, Arm_MI.Transform,
Arm_Orientation, Arm_Translation) ;

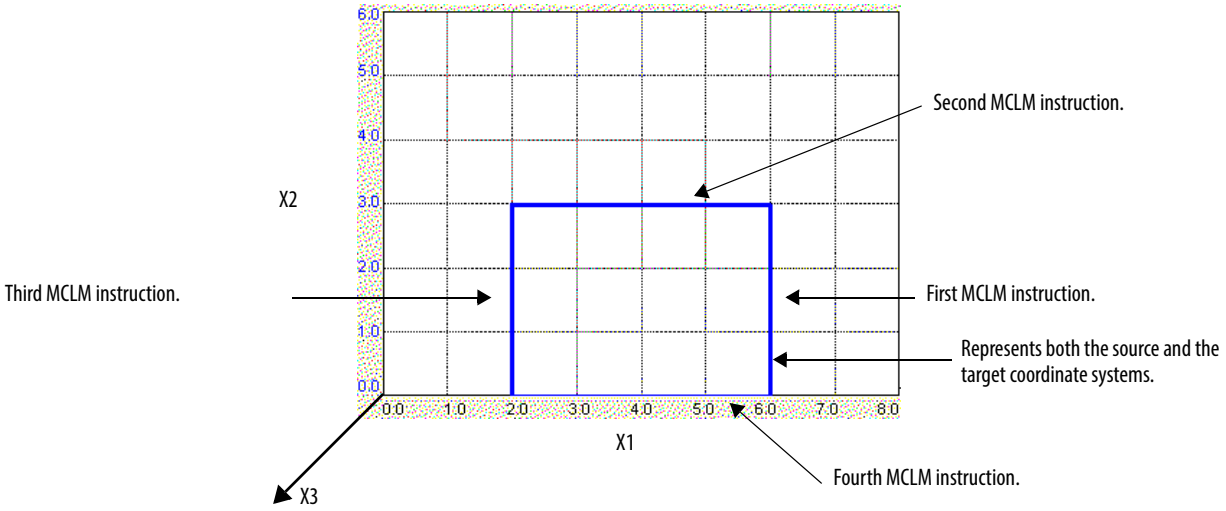
3. Pick and place routine

This routine is one in a sequence of MCLM instructions that move the Cartesian system. The joints of the robot follow the moves.


The SFC starts the pick and place moves when the Run bit turns on.
This step moves the coordinate system to 0, 6, 2. The P1 qualifier limits this to the first scan of the step.
When the move is in process (IP), the SFC goes to the next step and queues the next move.
Step_000
Tran_001
Arm_Commands.Run
Step_003
P1
Move_0
MCLM(Arm_X1_X2_X3,Arm_MI.Move[0],0,Arm_Positions[0,0],
Auto_Speed,Unitspersec,Cartesian_Vars.C.Auto_Accel,Unitspersec2,
Cartesian_Vars.C.Auto_Decel,Unitspersec2,Trapezoidal,1,Disabled,Programmed);
Tran_003
Arm_MI.Move[0].IP & NOT Arm_X1_X2_X3.MovePendingQueueFullStatus

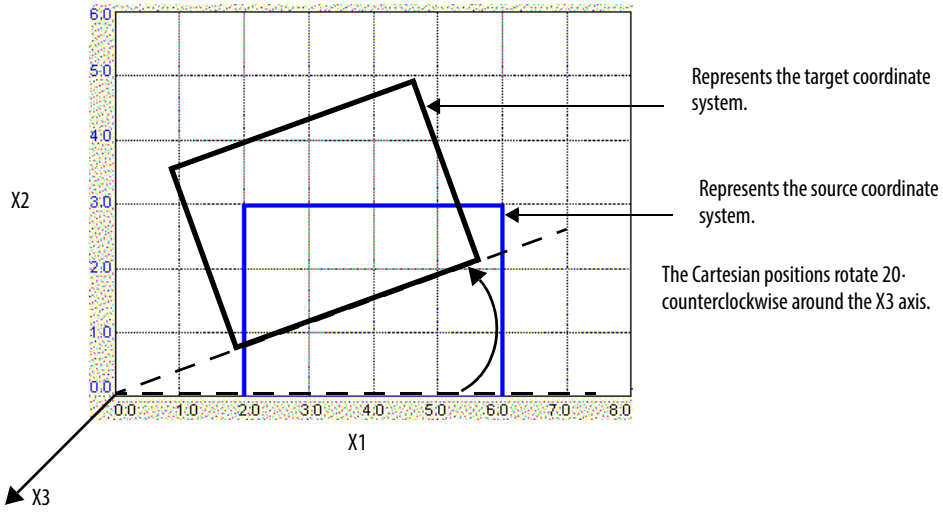
Example 3 - Change Orientation

If you want to move the target coordinate system in a rectangular path, execute the MCT instruction to start the transform. Then, execute four Motion Coordinated Linear Move (MCLM) instructions to produce the rectangular path.



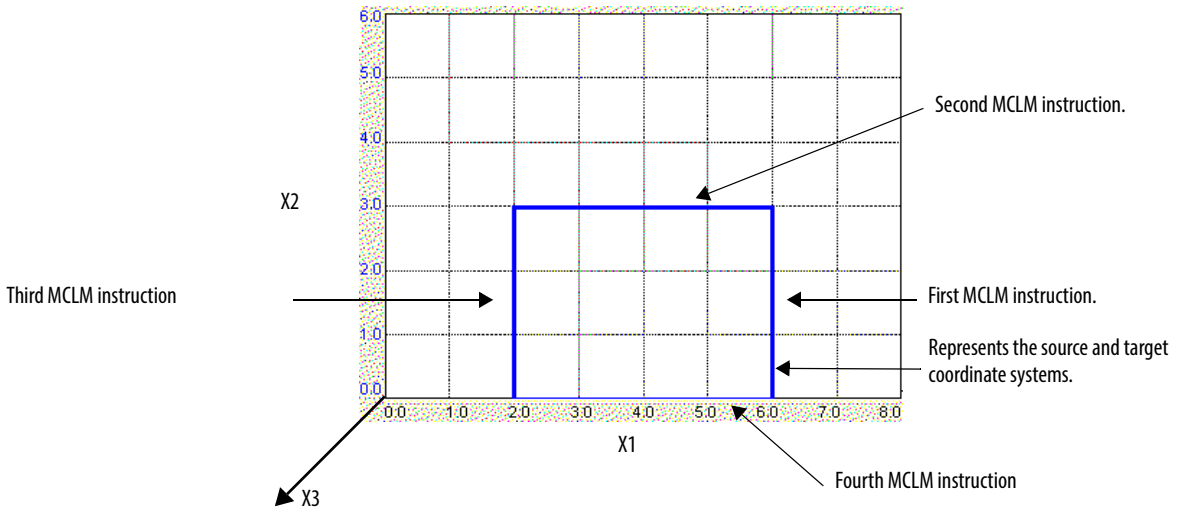
If you want to rotate the Cartesian positions of the target coordinate system by 20· counterclockwise around the X3 axis:

1. Enter orientation values of 0·, 0·, 20· into the MCT instruction.
2. Execute the MCT instruction again to apply the orientation to the transform.
3. Execute the same four MCLM instructions again.



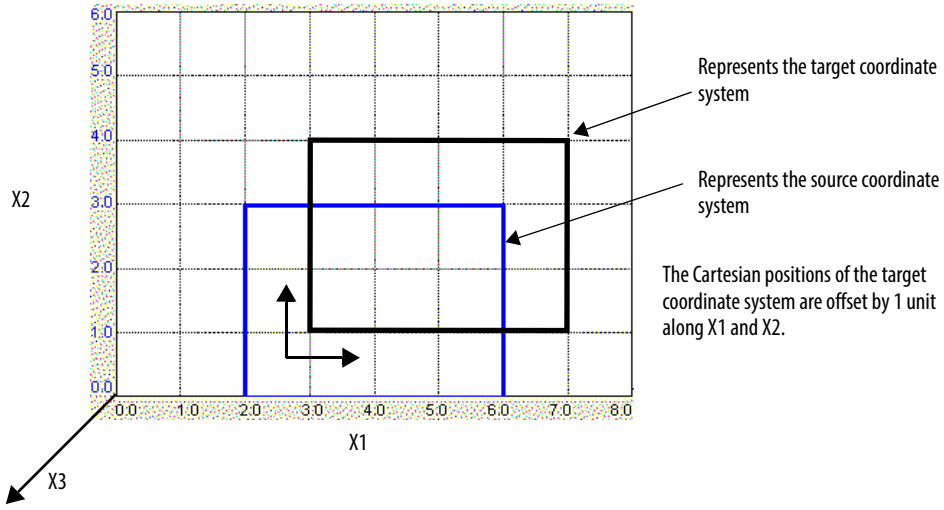
Example 4 - Change Translation

If you want to move the target coordinate system in a rectangular path, execute the MCT instruction to start the transform. Then, execute four Motion Coordinated Linear Move (MCLM) instructions to produce the rectangular path.



If you want to offset the Cartesian positions of the target coordinate system by 1 unit along both the X1 and X2 axes:

1. Enter translation values of 1, 1, 0 into the MCT instruction.
2. Execute the MCT instruction again to apply the translation to the transform.
3. Execute the same four MCLM instructions again.


Represents the target coordinate system
X2
Represents the source coordinate system
The Cartesian positions of the target coordinate system are offset by 1 unit along X1 and X2.
X1
X3

Motion Calculate Transform Position (MCTP)

Use the MCTP instruction to calculate the position of a point in one coordinate system to the equivalent point in a second coordinate system.

ATTENTION: Use each tag for the motion control attribute of instructions only once. Re-use of the motion control tag in other instructions can cause unintended operation. This can result in damage to equipment or personal injury.

IMPORTANT You can use this instruction with the following controllers:

- 1756-L6x controllers
- 1756-L7x controllers
- 1769-L18ERM controller
- 1769-L27ERM controller
- 1769-L30ERM controller
- 1769-L33ERM controller
- 1769-L36ERM controller

Operands

The MCTP instruction supports the following operands.

- Relay Ladder
- Structured Text

Relay Ladder

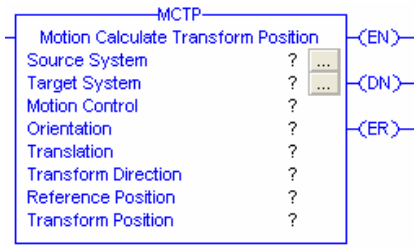


Table 52 - MCTP Instruction Operand Relay Ladder

Operand	Type	Format	Description
Source System	COORDINATE_SYSTEM	Tag	Cartesian coordinate system for Cartesian positions of the robot
Target System	COORDINATE_SYSTEM	Tag	Non-Cartesian coordinate system that controls the actual equipment
Motion Control	MOTION_INSTRUCTION	Tag	Control tag for the instruction

Table 52 - MCTP Instruction Operand Relay Ladder

<table>
<tr><th>Operand</th><th>Type</th><th>Format</th><th>Description</th></tr>
<tr><td rowspan="4">Orientation</td><td rowspan="4">REAL[3]</td><td rowspan="4">Array</td><td>Do you want to rotate the target position around the X1, X2, or X3 axis?
<table><tr><th>If</th><th>Then</th></tr><tr><td>No</td><td>Leave the array values at zero.</td></tr><tr><td>Yes</td><td>Enter the degrees of rotation into the array. Put the degrees of rotation around X1 in the first element of the array, and so on.</td></tr></table></td></tr>
<tr><td>Use an array of three REALs even if a coordinate system has only one or two axes.</td></tr>
<tr><td rowspan="4">Translation</td><td rowspan="4">REAL[3]</td><td rowspan="4">Array</td><td>Do you want to offset the target position along the X1, X2, or X3 axis?
<table><tr><th>If</th><th>Then</th></tr><tr><td>No</td><td>Leave the array values at zero.</td></tr><tr><td>Yes</td><td>Enter the offset distances into the array. Enter the offset distances in coordination units. Put the offset distance for X1 in the first element of the array, and so on.</td></tr></table></td></tr>
<tr><td>Use an array of three REALs even if a coordinate system has only one or two axes.</td></tr>
<tr><td>Transform Direction</td><td>DINT</td><td>Immediate</td><td><table><tr><th>For Robot Type</th><th>To calculate</th><th>With the base turned to the</th><th>And the robot is</th><th>Choose</th></tr><tr><td>All</td><td>Cartesian position</td><td>►</td><td>►</td><td>Forward</td></tr><tr><td>Cartesian
Delta 2D
Delta 3D
SCARA Delta</td><td>Joint angles</td><td>►</td><td>►</td><td>Inverse</td></tr><tr><td rowspan="4">Articulated Independent
Articulated Dependent
SCARA Independent</td><td rowspan="4">Joint angles</td><td rowspan="2">Same quadrant as the point</td><td>Right arm configuration</td><td>Inverse Right Arm</td></tr><tr><td>Left arm configuration</td><td>Inverse Left Arm</td></tr><tr><td rowspan="2">Opposite quadrant from the point</td><td>Right arm configuration</td><td>Inverse Right Arm Mirror</td></tr><tr><td>Left arm configuration</td><td>Inverse Left Arm Mirror</td></tr></table></td></tr>
<tr><td>Reference Position</td><td>REAL[3]
(units=coordination units)</td><td>Array</td><td><table><tr><th>If the transform direction is</th><th>Then enter an array that has the</th></tr><tr><td>Forward</td><td>Joint angles</td></tr><tr><td>Inverse</td><td>Cartesian positions</td></tr></table></td></tr>
<tr><td>Transform Position</td><td>REAL[3]
(units=coordination units)</td><td>Array</td><td>Array that stores the calculated position</td></tr>
</table>

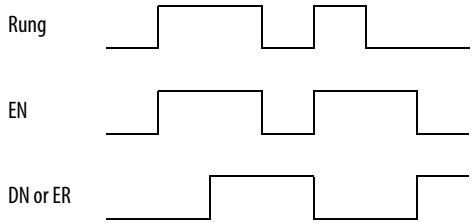
MCTP(Source System, Target System, Motion Control, Orientation, Translation, Transform Direction, Reference Position, Transform Position);

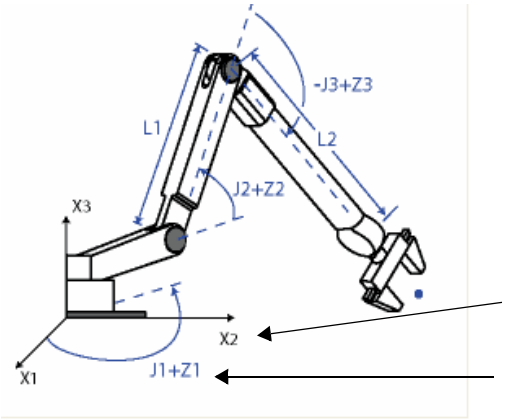
Structured Text

The structured text operands are the same as the ladder diagram operands. Enter the transform direction without spaces.

Example: Enter a transform direction of Inverse Left Arm as InverseLeftArm

Table 53 - Motion Instruction Data Type

To see if	Check if this bit is on	Data type	Notes
The rung is true.	EN	BOOL	Sometimes the EN bit stays on even if the rung goes false. This happens if the rung goes false before the instruction is done or an error has occurred. 
The instruction is done.	DN	BOOL	
An error happened.	ER	BOOL	Identify the error number listed in the error code field of the Motion control tag then, see Error Codes (ERR) for Coordinate Motion Instructions on page 261.



You can give the instruction the X1, X2, and X3 positions and get the corresponding J1, J2, and J3 angles.

Or you can give the instruction the J1, J2, and J3 angles and get the corresponding X1, X2, and X3 positions.

The MCTP instruction is similar to the MCT instruction except the MCTP instruction does not start a transform. It calculates a position once each time you execute it.

Programming Guidelines

Follow these guidelines to use an MCTP instruction.

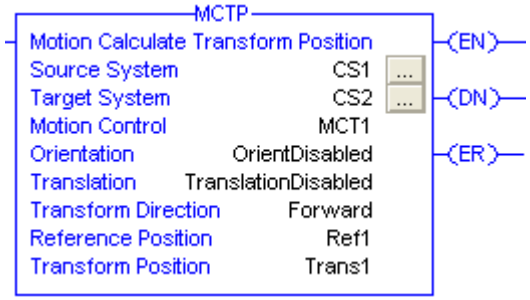
Table 54 - MCTP Programming Giuidelines and Examples

Guideline	Examples and notes
Toggle the rung from false to true to execute the instruction.	This is a transitional instruction. In a ladder diagram, toggle the rung-condition-in from false to true each time you want to execute the instruction.
In structured text, condition the instruction so that it only executes on a transition.	In structured text, instructions execute each time they are scanned. Condition the instruction so that it only executes on a transition. Use either of these methods: • Qualifier of an SFC action • Structured text construct

Data Flow of MCTP Instruction Between Two Coordinate Systems

The following illustrations show the flow of data when an MCTP Instruction is executed to perform a forward transformation and an inverse transformation. The CS1 indicator represents a Cartesian coordinate system containing X1, X2 and X3 axes as the source of the MCTP instruction. The CS2 indicator represents the joint coordinate system containing J1, J2 and J3 axes as the target of the MCTP instruction.

Figure 103 - Data Flow When a Move is Executed with an MCTP Instruction - Forward Transform



Input Data

CS1: DATA	SOURCE
Link Lengths (L1, L2)	Coordinate System dialog
Base Offsets (X1b, X2b, X3b)	Coordinate System dialog
End Effector Offsets (X1e, X2e, X3e)	Coordinate System dialog
Zero Angle Orientations (Z1, Z2, Z3)	Coordinate System dialog
Orientation (Array [3]) (Coordination Units)	Instruction Faceplate
Translations (Array [3]) Coordination Units)	Instruction Faceplate
Transform Direction	Instruction Faceplate
Reference Position Coordination Units) Typically Cartesian - Source Coordination Units)	Instruction Faceplate

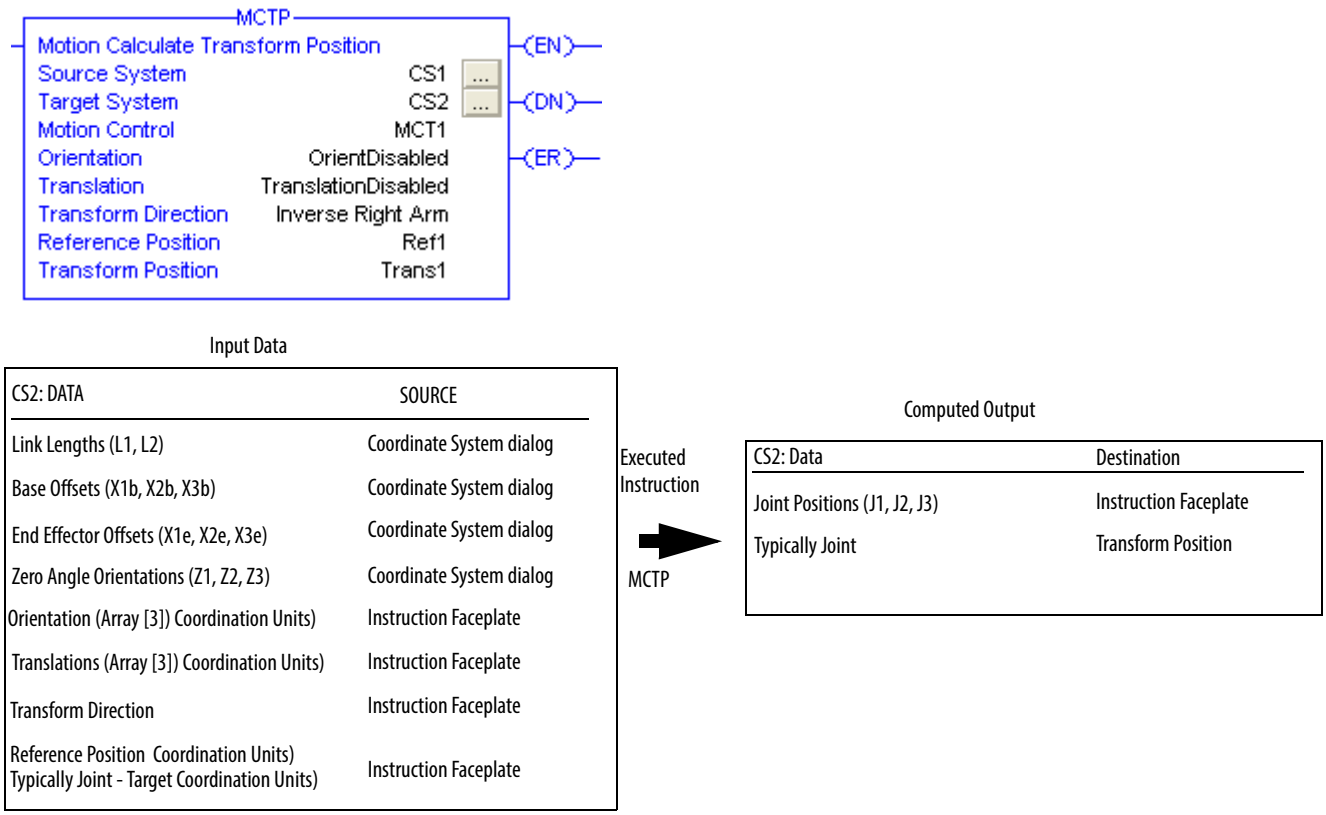
Executed Instruction

MCTP

Computed Output

CS1: Data	Destination
Cartesian Positions (X1, X2, X3)	Instruction Faceplate
Typically Cartesian	Transform Position

Figure 104 - Data Flow When a Move is Executed with an MCTP Instruction - Inverse Transform



Arithmetic Status Flags

Not affected.

Fault Conditions

None.

Error Codes

See Error Codes (ERR) for Coordinate Motion Instructions on page 261.

Extended Error Codes

Extended Error codes help to further define the error message given for this particular instruction. Their behavior is dependent upon the Error Code with which they are associated.

Table 55 - Error Code Corrective Actions

ERR	EXERR	Corrective Action	Notes
61	1	Assign both coordinate systems to the motion group.	
	2	Check that you are using the correct source and target systems.	You cannot use the same coordinate system as source and target.
	3	Set the transform dimension of the source system to the number of axes in the system, up to three.	
	4	Set the transform dimension of the target system to the number of axes in the system to be transformed, up to three.	
	5	Use a different source system.	You can only use one coordinate system as the source for one active transform.
	6	Use a different target system.	You can only use one coordinate system as the target for one active transform.
	7	Look for source or target axes that you are already using in another transform. Use different axes in the coordinate system.	You can only use an axis in one source system and one target system.
	8	Use a target system that isn't the source for this chain of transforms.	You cannot create a circular chain of transforms that leads back to the original source.
	9	Check that you've assigned the correct axes to each coordinate system.	You cannot use the same axes in the source and target systems.
	10	Stop all motion processes for all the axes in both systems (for example, jog, move, and gear).	You cannot start the transform if any motion process is controlling a source or target axis.
	11	Insufficient resources available to initiate the transform connection.	
	12	Set the link lengths.	You cannot use a link length of zero.
	13	Look for source or target axes that are in the shutdown state. Use a Motion Axis Shutdown Reset (MASR) instruction or direct command to reset the axes.	
	14	Uninhibit all the source or target axes.	
	15	Check the configured values for the base offsets and end effector offsets for the Delta or SCARA Delta robot.	(X1b-X1e) can not be less than 0.0 for both the Delta and SCARA Delta robots. For Delta robots, this error can also occur if the value of L1 + (X1b-X1e) is greater than L2.
	16	Check the SCARA independent and SCARA Delta robot configurations to be sure that: • the transform dimension for the source coordinate system is configured as 2. • the configured third axes for the source coordinate system and the target coordinate system are the same.	

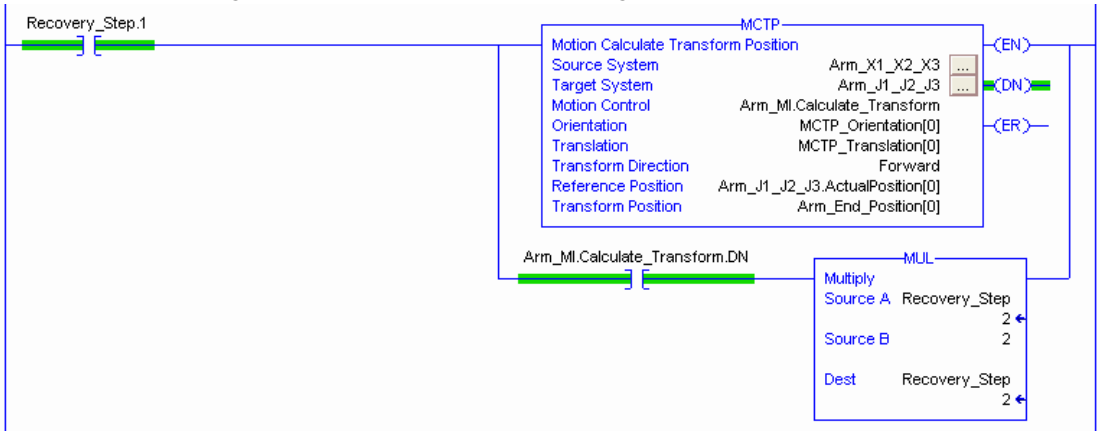
MCTP Changes to Status Bits

None.

Example 1 - Calculate Position

If you want to write a recovery sequence for faults, as one of your steps, you want to get the current position of an articulated independent robot. In that case, you can use an MCTP instruction to calculate the robot's Cartesian position when you know its joint angles.

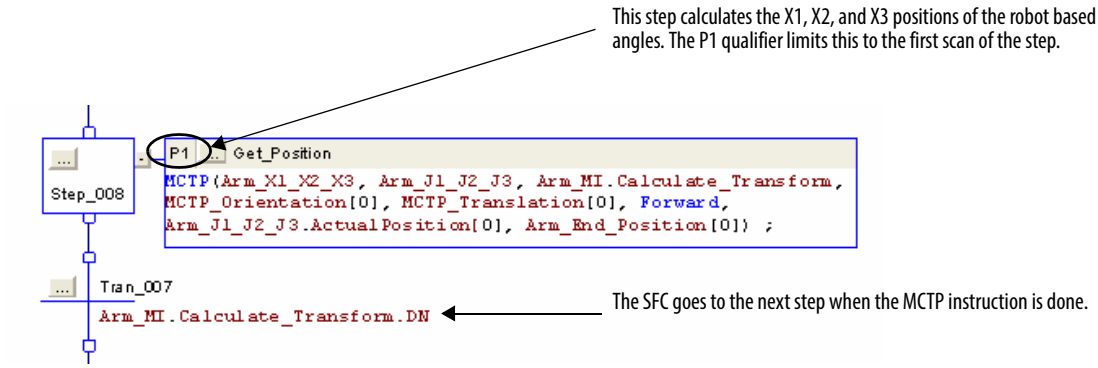
Figure 105 - Calculate Position—Ladder Diagram



If Recovery_Step.1 turns on, then calculate the X1, X2, and X3 positions of the robot based on its current joint angles

When the instruction is done, the MUL instruction takes the sequence to the next step

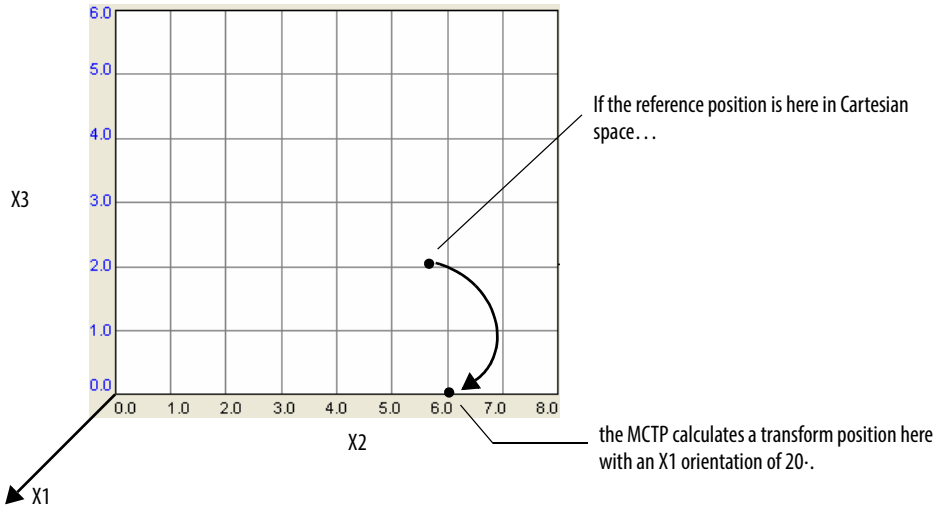
Figure 106 - Calculate Position - Structured Text


This step calculates the X1, X2, and X3 positions of the robot based on its current joint angles. The P1 qualifier limits this to the first scan of the step.
Step_008
P1
Get_Position
MCTP(Arm_X1_X2_X3, Arm_J1_J2_J3, Arm_MI.Calculate_Transform,
MCTP_Orientation[0], MCTP_Translation[0], Forward,
Arm_J1_J2_J3.ActualPosition[0], Arm_End_Position[0]) ;
Tran_007
Arm_MI.Calculate_Transform.DN
The SFC goes to the next step when the MCTP instruction is done.

Example 2 - Change Orientation

If you want to enter orientation values of 20·, 0·, 0· into example 1, the MCTP instruction does a forward transform.

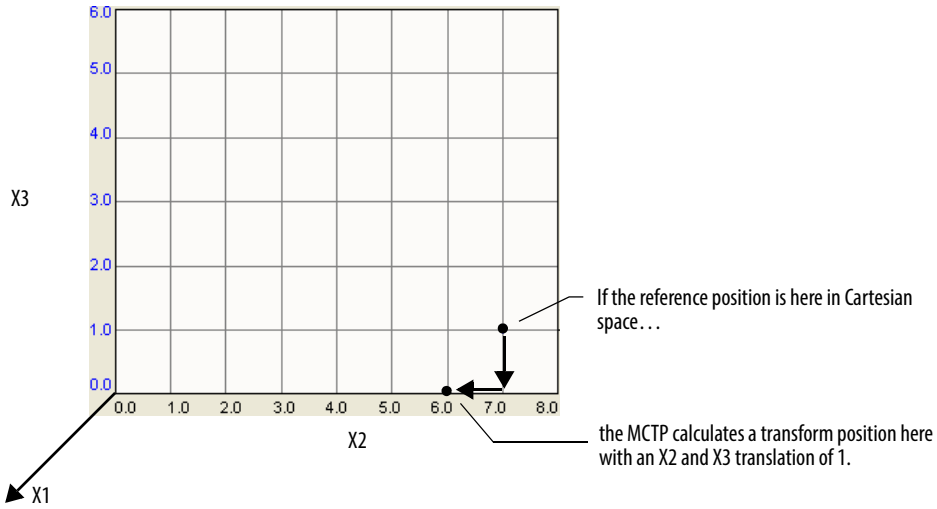
Figure 107 - Change Orientation



Example 3- Change Translation

If you want to enter translation values of 0, 1, 1 into example 1, the MCTP instruction does a forward transform.

Figure 108 - Change Translation



Example 4 - Change Direction

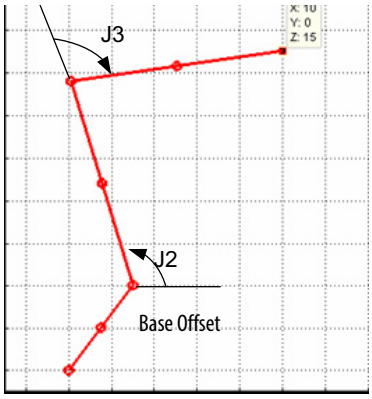
If your robot has base offsets, there can be as many as four ways to get to a given point. If your robot has this geometry:

- L1 = 10.
- L2 = 10.
- X1b = 3.0.
- X3b = 4.0.

This example shows the ways to get a position of X1 = 10, X2 = 0, and X3 = 15

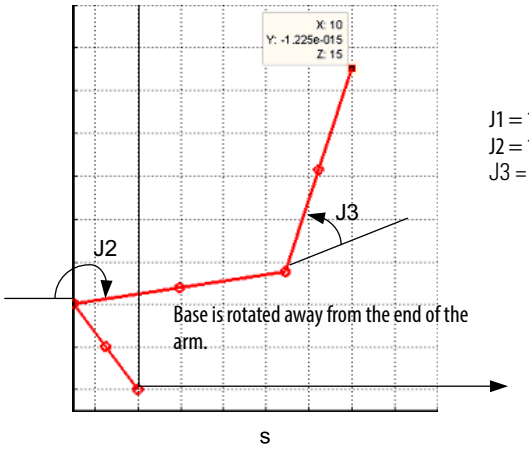
Figure 109 - Transform Direction

Inverse left arm



J1 = 0
J2 = 106.84
J3 = -98.63

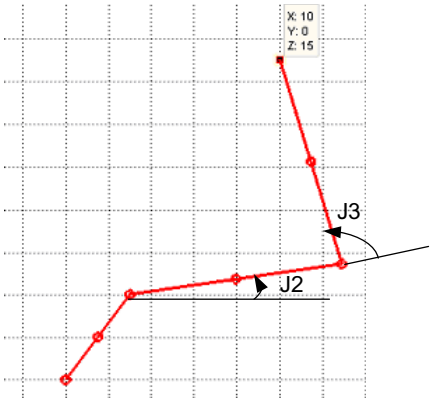
Inverse left arm mirror



J1 = 180
J2 = 171.39
J3 = -63.26

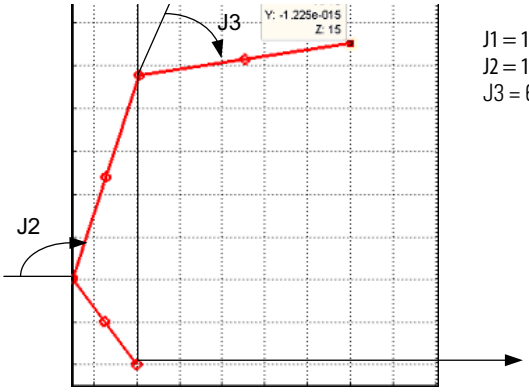
s

Inverse right arm



J1 = 0
J2 = 8.22
J3 = 98.63

Inverse right arm mirror



J1 = 180
J2 = 108.14
J3 = 63.26

Motion Coordinated Shutdown Reset (MCSR)

Use the Motion Coordinated Shutdown Reset (MCSR) instruction to reset all axes in a coordinate system. The MCSR instruction resets the axes from a shutdown state to an axis ready state. This instruction also clears any axis faults.

ATTENTION: Use each tag for the motion control attribute of instructions only once. Re-use of the motion control tag in other instructions can cause unintended operation. This can result in damage to equipment or personal injury.

Operands

The MCSR instruction supports the following operands.

- Coordinate System
- Motion Control

Relay Ladder

Table 56 - MCSR Instruction Operands Relay Ladder

Operand	Type	Format	Description
Coordinate System	COORDINATE_SYSTEM	Tag	The Coordinate System operand specifies the set of motion axes that define the dimensions of a Cartesian coordinate system. The coordinate system supports up to three primary axes. Only the axes configured as primary axes (up to three) are included in the coordinate velocity calculations. Name of the axis, which provides the position input to the Output Cam.
Motion Control	MOTION_INSTRUCTION	Tag	Structure used to access instruction status parameters.

Structured Text

```
MCSR(CoordinateSystem,
MotionControl);
```

The operands are the same as those for the relay ladder MCSR instruction.

Executing the Instruction

This is a transitional instruction.

- In relay ladder, toggle the rung-condition-in from cleared to set each time the instruction executes.
- In structured text, condition the instruction so that it only executes on a transition.

Motion Control Bits

The following control bits are affected by the MCSR instruction.

Table 57 - Control Bits Affected by MCSR Instruction

Mnemonic	Description
.EN (Enable) Bit 31	The Enable bit sets when the rung transitions from false to true. It resets when the rung goes from true to false.
.DN (Done) Bit 29	The Done bit sets when the coordinated shutdown reset is successfully initiated. It resets when the rung transitions from true to false.
.ER (Error) Bit 28	The Error bit sets when the reset of the coordinated shutdown fails to initiate. It resets when the rung transitions from false to true.

Arithmetic Status Flags

Not affected.

Fault Conditions

None.

Error Codes

See Error Codes (ERR) for Coordinate Motion Instructions on page 261.

MCSR Changes to Status Bits

Status Bits provide a means for monitoring the progress of the motion instruction. There are three types of Status bits that provide pertinent information. They are Axis Status bits, Coordinate System Status bits, and Coordinate Motion Status bits. When the MCS instruction initiates, the status bits undergo the following changes.

Table 58 - Axis Status Bits

Bit Name	Effect
CoordinatedMoveStatus	No effect.

Table 59 - Coordinate System Status Bits

Bit Name	Effect
ShutdownStatus	Clears the Shutdown status bit.

Table 60 - Coordinate Motion Status Bits

Bit Name	Effect
MovePendingStatus	Flushes instruction queue and clears status bit.
MovePendingQueueFullStatus	Flushes instruction queue and clears status bit.

Figure 110 - Relay Ladder Example

MCSR(CoordinatedSyst,MCSR[3];

Structured Text

MCSR(Coordinated_sys,MCSR[3]);

Master Driven Coordinate Control (MDCC)

Use the MDCC instruction to synchronize one or more motion axes or Coordinate System to a common Master Axis. The Master Driven Speed Control (MDSC) function uses the Master Driven Coordinated Control (MDCC) instruction, which defines a Master:Slave relationship between a Master Axis and a Slave Coordinate System.

For information about the Master:Slave relationship for single axes, see the Logix5000 Controllers Motion Instructions Reference Manual, publication MOTION-RM002.

The Motion Master Driven Coordinate Control instruction (MDCC) is used when the Slave System is a Coordinate System.

The MDCC instruction defines the relationship between the external Master Axis and the Slave Coordinate System for the MCLM and the MCCM Instructions. When an MDCC is executed (goes IP), the specified Slave Coordinate System in the MDCC instruction is logically geared with the designated Master Axis. After motion on the Master Axis is initiated, all the axes in the Coordinate System specified as the Slave Coordinate System follow the Master Axis motion at the programmed dynamics of the programmed instruction.

There are no changes in any active motion when a new MDCC instruction is activated. Activating an MDCC instruction just puts the parameters programmed in the MDCC instruction into a pending state. The parameters in the pending MDCC instruction are changed if you execute a succeeding MDCC instruction before a new MCLM or MCCM instruction is activated. The MDCC becomes active (AC bit is set) only when all queued motion is complete and the motion queue is empty.

All motion in the queue keeps using the same Master Axis even if there is a pending MDCC with a different master. The values in the pending MDCC instruction are only used when:

- the next MCLM or MCCM instruction is activated on the Slave Coordinate System when the queue is empty, or
- an MCLM or MCCM is executed (goes IP) with a Merge type of All or Coordinate. (Note that this is because the merge empties the queue.)

Operands

The MDCC instruction supports the following operands:

- Relay Ladder
- Structured Text

Relay Ladder

Table 61 - Operands - Relay Ladder

Operand	Type	Format	Description
Slave System	COORDINATE_SYSTEM	Tag	The Coordinate System that the Master Axis controls when the motion planner is in Master Driven mode. Ellipsis launches the Coordinate System properties dialog. Upon verification, you receive a verification error if the Slave is a non-Cartesian Coordinate System or if the Master Axis is in the Slave Coordinate System. The MDCC link is broken when the following instructions are executed: • On any axis in the Slave Coordinate System or the Slave Coordinate System: MAS (All), MCS (All), MGS, MASD, MCSD, MGSD, a mode change. Note that MAS (anything other than All) and MCS do NOT break the MDCC link. The Shutdown instructions (MGSD, MASD, MCSD) never go IP. • On the Master Axis: MASD, MCSD, and MGSD. Note that MAS and MCS for any Stop Type, including All, do NOT break the MDCC link. A mode change (Rem Run to Rem Prog or Rem Prog to Rem Run) or an axis fault also breaks the MDCC link.
Master Axis	AXIS_CONSUMED AXIS_SERVO AXIS_SERVO_DRIVE AXIS_GENERIC AXIS_GENERIC_DRIVE AXIS_CIP_DRIVE AXIS_VIRTUAL	Tag	Any configured Single Axis that the Slave Coordinate System follows. The Master Axis can be any axis that has been configured.
Motion Control	MOTION_INSTRUCTION	Tag	Control tag for the instruction.
Master Reference	UNIT32	Immediate Tag	Selects the Master position source as either Actual Position (0) or Command Position (1).

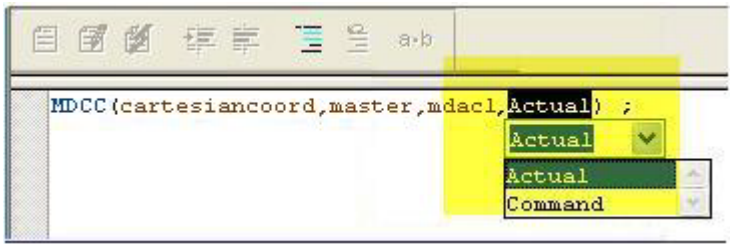
MDCC(Cartesian Coord, MasterAxis,MDSCl,Actual);

Structured Text

The operands for structured text are the same as those for the relay ladder MDCC instruction.

Note that you have the option to browse for enumerations in the Structured Text Editor as shown below.

Figure 111 - Enumerations in the Structured Text Editor



Master Reference

The Master Reference for an MDCC instruction selects the Master Axis position source.

The enumerations for Master Reference Axis are:

- Actual – Slave motion is generated from the actual (current) position of the Master Axis as measured by its encoder or other feedback device.
- Command – Slave motion is generated from the command (desired) position of the Master Axis.

Because there is no Command Position for a Feedback Only Axis, if you select either Actual or Command for Master Reference, the Actual Position of the Master Axis is used. The Actual and Command Position are always the same for this case. No error is generated.

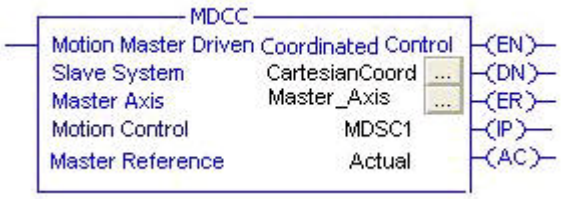
Because there is no Actual Position for a Virtual axis, if you select either Actual or Command for Master Reference, the Command Position is used. No error is generated.

An error is generated if a MDCC instruction is executed that changes the Master Reference of a Slave Coordinate System that is in motion. The new MDCC instruction creates an error and the original one remains active.

Motion Direct Command and the MDCC Instruction

To obtain Motion Direct support for the MDCC instruction, you must first program an MDCC in one of the supporting program languages before you execute an MCLM or MCCM in Time driven Mode.

Ladder Diagram



Structured Text

MDCC (Cartesian Coord, Master Axis, MDSC1, Actual);

In the above examples:

Parameter	Definition
CartesianCoord	CartesianCoord is the Coordinate System that is being controlled by the Master Axis when the motion planner is in Master Driven Mode.
Master Axis	Master Axis is the single axis that the Slave Coordinate System follow.
MDSC1	MDSC1 is the control tag for the MDCC instruction.
Actual	Actual Position is the position source of the Master Axis.

MOTION_INSTRUCTION Bit Leg Definitions for MDCC

Mnemonic	Description
.EN (Enable) Bit 31	The enable bit is set when the rung transitions from false-to-true and stays set until the rung goes false.
.DN (Done) Bit 29	The done bit is set when the coordinate MDCC instruction is successfully initiated.
.ER (Error) Bit 28	The error bit is set when there is an invalid combination of parameters in the MDCC instruction.
.IP (In Process) Bit 26	The in process bit is set when the MDCC instruction is activated and reset by an instruction (for example, the MCSD instruction).
.AC (Active) Bit 23	The active bit is set when an MCLM or MCCM is activated (that is, when the AC bit of the MCLM or MCCM instruction is set) on a Coordinate System that is selected as the Slave Coordinate System of the MDCC instruction.

Arithmetic Status Flags

Not affected

Fault Conditions for Motion Instructions when MDCC Is Active

All commands in the following table are for the Slave Coordinate System.

Table 62 - MDCC Fault Conditions

Instruction	Parameters	MDCC IP Bit
MGS		Reset
MGSD		Reset
MCS	Stop Type = Coordinated Motion	Not Changed
	Stop Type = Transform	Not Changed
	Stop Type = All	Reset
MCSD		Reset
MAS	Stop Type = Jog	Not Changed
	Stop Type = Move	Not Changed
	Stop Type = Time CAM	Not Changed
	Stop Type = All	Reset
MASD		Reset
MSF		Not Changed
MDF		Not Changed
Fault Action	Status Only	Not Changed
	Stop Motion	Reset
	Disable DRV	Reset
	Shutdown	Reset

Note that if the same Slave Coordinate System is controlled by multiple Master Axes, if one MDCC relationship that contains the Slave Coordinate System is broken, then all MDCC relationships that contain the Slave Coordinate System are broken.

Common Action Table for Master Axis

All commands in the following table are for the Master Axis.

Table 63 - Master Axis Commands

Instruction	Parameters	MDCC IP Bit
MGS		Reset
MGSD		Reset
MCS	Stop Type = Coordinated Motion	Not Changed
	Stop Type = Transform	Not Changed
	Stop Type = All	Not Changed
MCSD		Reset
MAS	Any Stop Type (Jog, Move, Time CAM, All)	Not Changed

Table 63 - Master Axis Commands

MASD		Reset
MSF		Not Changed
MDF		Not Changed
Fault Action	Status Only	Not Changed
	Stop Motion	Not Changed
	Disable DRV	Not Changed
	Shutdown	Reset

Note that if the same Master Axis is controlling multiple Slave Coordinate System, then all MDCC relationships that contain the Master Axis are broken.

Error Codes

See Error Codes (ERR) for Coordinate Motion Instructions on page 261.

Logix Designer Application Verification Errors

An invalid or No Master Axis causes new errors to be generated when verified by Logix Designer application. The following conditions can cause this error:

- The Master Axis is a member of the Slave Coordinate System.
- The Master Axis or the Slave Coordinate System is not configured.
- The Master Axis or an axis in the Slave Coordinate System is inhibited.
- A redefine position is in progress.
- Home of the Master axis in the Slave Coordinate System is in progress.

Status Bits for Motion Instructions (MCLM, MCCM) when MDCC Is Active

The following table describes the predefined data type status bits for motion instructions MCLM and MCCM.

Table 64 - MCLM and MCCM Motion Instruction Bits

Bit Name	Meaning
EN	The Enable bit is set when the rung transitions from false to true and resets when the rung goes from true to false.
DN	The Done bit sets when the coordinated instruction has been verified and queued successfully. Because it is set at the time it is queued, it can appear as set when a runtime error is encountered during the verify operation after it comes out of the queue. It resets when the rung transitions from false to true.
ER	The Error bit is reset when the rung transitions from false to true. It is set when the coordinated move has not successfully initiated. It is also set with the Done Bit when a queued instruction encounters a runtime error.
PC	The Process Complete bit is reset when the rung transitions from false to true. It is set when there is no succeeding move and the coordinated move reaches the new position, or when there is a succeeding move and the coordinated move reaches the specified termination type.

Table 64 - MCLM and MCCM Motion Instruction Bits

IP	The In Process bit is set when the coordinated move is successfully initiated. It is reset when: • there is no succeeding move and the coordinated move reaches the new position, or • when there is a succeeding move and the coordinated move reaches the specifications of the termination type, or • when the coordinated move is superseded by another MCLM or MCCM instruction with a merge type of Coordinated Move, or • when terminated by an MCS instruction.
AC	When you have a coordinated move instruction queued, the Active bit lets you know which instruction is controlling the motion. It sets when the coordinated move becomes active. It is reset when the Process Complete bit is set or when the instruction is stopped.
ACCEL	Set as expected during motion. It is independent of Master acceleration. The ACCEL bit on the instruction driving the Slave Coordinate System (for example, MCLM) is set as the Slave Coordinate System is accelerating to its commanded speed. This bit is insensitive to acceleration occurring on the Master Axis. However, the AccelStatus bit, which is in the MotionStatus word of the Slave Coordinate System (not the instruction driving the Slave Coordinate System), is set or cleared based on changes in velocity of the Slave Coordinate System.
DECEL	Set as expected during motion. It is independent of Master deceleration. The DECEL bit on the instruction driving the Slave Coordinate System is set as the Slave Coordinate System is decelerating to its commanded speed. This bit is insensitive to deceleration occurring on the Master Axis. However, the DecelStatus bit, which is in the MotionStatus word of the Slave Coordinate System (not the instruction driving the slave axis), is set or cleared based on changes in velocity of the Slave Coordinate System.
TrackingMaster	Indicates that the Slave Coordinate System is tracking the Master Axis (only used in Master Driven Mode). When an instruction is initiated in Master Driven Mode, the Slave Coordinate System accelerates to the speed that is programmed for MDSC mode. The Tracking Master is set when the acceleration is complete in MDSC Mode. This means that the Slave Coordinate System is synchronized to the Master Axis. The Tracking Master bit is cleared when any of the following occurs on the Slave Coordinate System: • When the Slave Coordinate System starts to either accelerate or decelerate for any reason, for example, for an MCCD or an MCS being issued. • When the Slave Coordinate System is relinked to another Master Axis. In this situation, the TrackingMaster bit is first cleared and then it is set again in the new instruction status word when the Slave Coordinate System starts tracking the new Master Axis again. • The Slave Coordinate System is stopped. The Tracking Master is cleared as soon as the stop is initiated on the Slave Coordinate System. This bit is never set when LockDir = NONE. Note that the Tracking Master bit on the Slave Coordinate System is not affected by any operation (for example, MCS, MCCD) on the Master Axis. The Tracking Master bit is always cleared in Time Driven Mode.

Table 64 - MCLM and MCCM Motion Instruction Bits

IP	The In Process bit is set when the coordinated move is successfully initiated. It is reset when: • there is no succeeding move and the coordinated move reaches the new position, or • when there is a succeeding move and the coordinated move reaches the specifications of the termination type, or • when the coordinated move is superseded by another MCLM or MCCM instruction with a merge type of Coordinated Move, or • when terminated by an MCS instruction.
AC	When you have a coordinated move instruction queued, the Active bit lets you know which instruction is controlling the motion. It sets when the coordinated move becomes active. It is reset when the Process Complete bit is set or when the instruction is stopped.
ACCEL	Set as expected during motion. It is independent of Master acceleration. The ACCEL bit on the instruction driving the Slave Coordinate System (for example, MCLM) is set as the Slave Coordinate System is accelerating to its commanded speed. This bit is insensitive to acceleration occurring on the Master Axis. However, the AccelStatus bit, which is in the MotionStatus word of the Slave Coordinate System (not the instruction driving the Slave Coordinate System), is set or cleared based on changes in velocity of the Slave Coordinate System.
DECEL	Set as expected during motion. It is independent of Master deceleration. The DECEL bit on the instruction driving the Slave Coordinate System is set as the Slave Coordinate System is decelerating to its commanded speed. This bit is insensitive to deceleration occurring on the Master Axis. However, the DecelStatus bit, which is in the MotionStatus word of the Slave Coordinate System (not the instruction driving the slave axis), is set or cleared based on changes in velocity of the Slave Coordinate System.
TrackingMaster	Indicates that the Slave Coordinate System is tracking the Master Axis (only used in Master Driven Mode). When an instruction is initiated in Master Driven Mode, the Slave Coordinate System accelerates to the speed that is programmed for MDSC mode. The Tracking Master is set when the acceleration is complete in MDSC Mode. This means that the Slave Coordinate System is synchronized to the Master Axis. The Tracking Master bit is cleared when any of the following occurs on the Slave Coordinate System: • When the Slave Coordinate System starts to either accelerate or decelerate for any reason, for example, for an MCCD or an MCS being issued. • When the Slave Coordinate System is relinked to another Master Axis. In this situation, the TrackingMaster bit is first cleared and then it is set again in the new instruction status word when the Slave Coordinate System starts tracking the new Master Axis again. • The Slave Coordinate System is stopped. The Tracking Master is cleared as soon as the stop is initiated on the Slave Coordinate System. This bit is never set when LockDir = NONE. Note that the Tracking Master bit on the Slave Coordinate System is not affected by any operation (for example, MCS, MCCD) on the Master Axis. The Tracking Master bit is always cleared in Time Driven Mode.

Table 64 - MCLM and MCCM Motion Instruction Bits

CalculatedDataAvailable	Indicates that the requested data has been returned in the Calculated Data array element and that Logix Designer application has updated the output data in the Calculated Data parameter. Only one status bit is used to indicate all Calculated Data is available. For the CalculatedDataAvailabe status bit, the moves in the motion queue are processed in batches. The first batch in the motion queue includes all moves in the queue up to and including the first move with a term type TT0 or TT1, or a move with a speed of 0. For moves in either Time Driven mode or Mater Driven mode, the CalculatedDataAvailable bit is set when: • MCLM or MCCM is enqueued and belongs to the first batch in the queue. There are two exceptions: – Moves with a speed of 0, although belonging to the first batch, do not have their CalculatedDataAvailable bit set. Their CalculatedDataAvailable bit is set after their Speed is changed to nonzero with a MCCD. – Moves with a term type TT2 through TT6 do not have their CalculatedDataAvailable bit set if they are the last move in the queue. CalculatedDataAvailable bit is cleared by: • MAS (all) or MASD - This clears the CalculatedDataAvailable bit of the active MAMs and all enqueued MCLM or MCCMs that contain the specified axis. • MCS (coordinated) - This only clears the CalculatedDataAvailable bit for all enqueued MCLM or MCCMs in the coordinate system being stopped. • MCS (all) or MCSD - This clears the CalculatedDataAvailable bit of all active MAMs that contain any axes in the referenced coordinate system and all enqueued MCLM or MCCMs of the coordinate system being stopped. • MGS or MGSD is executed (goes IP) - This clears the CalculatedDataAvailable bit of all active MAMs and all enqueued MCLM or MCCMs of the group being stopped or shutdown. • MCD or MCCD is executed (goes IP) - The CalculatedDataAvailable bit is reset and is immediately set again. • A MCLM or MCCM is executed (goes IP) with a merge enabled (either Coordinated or Merge All) - The CalculatedDataAvailable bit of all enqueued MCLM or MCCMs are cleared. MCLMs and MCCMs that are blending with the next coordinated motion instruction are still considered to be enqueued even if their PC flag was set when the blending was started. The CalculatedDataAvailable bit is not set for any move that Event Distance is not specified (that is, for any move where the Event Distance parameter in the instruction is zero). MSF and MDF do not alter the state of the CalculatedDataAvailable bit.

Coordinated Motion Status Bits

Bit Name	Meaning
CoordinateMotionStatus	Set when an axis lock is requested for an MCLM or MCCM instruction and the axis has crossed the Lock Position. Cleared when an MCLM or MCCM is initiated.
AccelStatus	Sets when vector is accelerating. Clears when a blend is in process or when vector move is at speed or decelerating.
DecelStatus	Sets when vector is decelerating. Clears when a blend is in process or when vector move is accelerating or when move completes.
ActualPosToleranceStatus	Sets for Actual Tolerance termination type only. The bit is set after the following two conditions have been met. 1) Interpolation is complete. 2) The actual distance to the programmed endpoint is less than the configured coordinate system's Actual Tolerance value. It remains set after the instruction completes. It is reset when a new instruction is started.
CommandPosToleranceStatus	Sets for all termination types whenever the distance to the programmed endpoint is less than the configured coordinate system's Command Tolerance value and remains set after the instruction completes. It is reset when a new instruction is started.
StoppingStatus	The Stopping Status bit is cleared when the MCCM instruction executes.
MoveStatus	Sets when MCCM begins axis motion. Clears on the .PC bit of the last motion instruction or a motion instruction executes that causes a stop.
MoveTransitionStatus	Sets when No Decel or Command Tolerance termination type is satisfied. When blending collinear moves, the bit is not set because the machine is always on path. It clears when a blend completes, the motion of a pending instruction starts, or a motion instruction executes that causes a stop. Indicates not on path.
MovePendingQueueFullStatus	Sets when the instruction queue is full. It clears when the queue has room to hold another new coordinated move instruction.
TransformSourceStatus	The coordinate system is the source of an active transform.
TransformTargetStatus	The coordinate system is the target of an active transform.
CoorMotionLockStatus	Set when an axis lock is requested for an MCLM or MCCM instruction and the axis has crossed the Lock Position. Cleared when an MCLM or MCCM is initiated. For the enumerations Immediate Forward Only and Immediate Reverse Only, the bit is set immediately when the MCLM or MCCM is initiated. When the enumeration is Position Forward Only or Position Reverse Only, the bit is set when the Master Axis crosses the Lock Position in the specified direction. The bit is never set if the enumeration is NONE. The CoordMotionLockStatus bit is cleared when the Master Axis reverses direction and the Slave Coordinate System stops following the Master Axis. The CoordMotionLockStatus bit is set again when the Slave Coordinate System resumes following the Master Axis. The CoordMotionLockStatus bit is also cleared when an MCS is initiated.

Changing Between Master Driven and Time Driven Modes for Coordinated Motion Instructions

Changing the motion mode between Master Driven and Time Driven Mode and vice versa is automatically performed when another motion instruction (such as, MCLM and MCCM) is activated if the new instruction has been programmed in a different mode than the active motion instruction.

When the new motion instruction is activated, the system assumes that the desired mode for the new instruction is the mode (Master Driven or Time Driven) as specified in the programmed units of the speed parameter contained in the new instruction. At all times, including when changing modes, the Accel, Decel, and Jerk must be programmed in the same units as the Speed parameter or the instruction gets a MDSC_UNITS_CONFLICT_ERROR error.

A runtime MDSC_INVALID_MODE_OR_MASTER_CHANGE error occurs if you attempt to change from Master Driven Mode to Time Driven Mode or vice versa with an MCCD instruction.

If both the master and slave axes are idle (for example, paused), the MCLM or MCCM can make a change on the slave. However, the error MDSC IDLE_MASTER_AND_SLAVE_MOVING is generated if MDSC mode is started while the Slave Coordinate System is moving when the master is idle.

Different Time Driven and Master Driven Modes can be used for different motion types for superimposed motion. For example, the MAM can be in time drive mode for an axis in the Coordinate System and the MCLM can be in Master Driven Mode for the Coordinate System.

Changing the Master Axis

The following sequence of events must be followed to transfer a Slave Coordinate System from one Master Axis to a second Master Axis.

- First, you must execute an MDCC instruction to reassign the Slave Coordinate System from the first Master Axis to the second Master Axis. This makes the reassignment pending. The IP bit of the MDCC instruction is set as an indication of the pending reassignment.

- Second, you must execute a new motion command (for example, an MCLM or MCCM). The Slave Coordinate System becomes unlocked from the first Master Axis and reassigned to the second Master Axis when this motion instruction is executed (goes IP).

The final effective slave speed is computed as the product of the Master Axis' speed and the slave's programmed speed. If the new final effective Slave Coordinate System speed is less than 10%, depending on the move of the original Slave Coordinate System speed, then the change is not allowed and the MDSC_INVALID_SLAVE_SPEED_REDUCTION error occurs. If the second Master Axis is idle (velocity=0), the motion instruction making this request receives an MDSC_IDLE_MASTER_AND_SLAVE_MOVING error.

If the second Master Axis is moving while the transfer is being made, then you can look at the TrackingMaster instruction status bit of the motion instruction that is performing the transfer to determine when the transfer is finished. This bit is set when the acceleration or deceleration on the Slave Coordinate System is

complete. At which time the Slave Coordinate System is synchronized to the second Master Axis.

Input and Output Parameters Structure for Coordinate System Motion Instructions

The middle column of the table below identifies which parameter is applicable to each coordinate system motion instruction, that is, to MCLM and MCCM. Before any of the parameters identified in the first column below can be used in the MCLM or MCCM instruction, you must execute an MDCC instruction and it must be active (IP bit is set).

Table 65 - Coordinate System Input and Output Parameters

Parameter	Instruction	Mode
Input Parameters		
Lock Direction	MCLM, MCCM	Master Driven Only
Lock Position	MCLM, MCCM	Master Driven Only
Command Tolerance	MCLM, MCCM	Master Driven and Time Driven
Event Distance	MCLM, MCCM	All modes (Master Driven or Time Driven)
Output Parameter		
Calculated Data	MCLM, MCCM	All modes (Master Driven, Time Driven, and Timed Based)

The following table describes the input parameters.

Table 66 - Input Parameters

<table>
<tr><th>Input Parameter</th><th>Data Type</th><th colspan="3">Description</th><th>Valid and Default Values</th></tr>
<tr><td>Lock Direction</td><td>IMMEDIATE</td><td colspan="3">This parameter is used for both Time Driven and Master Driven Mode. The controlling axis is the Master Axis (axis is programmed in the MDCC command) for Master Driven Mode and the axis that is programmed in the motion instruction (for example, MCLM) for Time Driven Mode.
The first word of the Lock Direction enumeration definition (see enumeration table below) identifies the lock type as either:
• Immediate (lock is performed immediately), or
• Position (lock is performed when the Master Axis crosses the specified Lock Position).
The second word of the enumeration specifies the direction in which the Master Axis has to be moving when it crosses the Lock Position for the lock to take effect.
For an MCLM and MCCM instruction, the Slave Coordinate System always moves in one direction - its programmed direction - while it follows the Master Axis, regardless of the direction of the Master Axis. If the Master reverses, the Slave Coordinate System stops.
For Master Driven Mode, the enumerations are as follows:
(Forward is positive velocity, reverse is negative velocity.)
The enumerations table is below.
Enumeration | Definition | Description
0 | None | Indicates that the Lock Position is not active.
If Lock Direction is set to None and the Master Driven mode is selected by the speed parameter of the motion instruction, the system will error.
Conversely, if Lock direction is not set to a value other than None and the speed parameter units indicate Time Driven mode, an error is also generated.
1 | Immediate Forward Only | Motion starts immediately when the Master is moving in the Forward Direction. The Master Axis is only followed while it is moving in the Forward Direction.
2 | Immediate Reverse Only | Motion starts immediately when the Master Axis is moving in the Reverse Direction. The Master Axis is only followed while it is moving in the Reverse Direction.</td><td>Valid = 0…4
Default = None
(Enumeration 1…4 are currently not allowed in Time Driven mode.)</td></tr>
</table>

Table 66 - Input Parameters

<table>
<tr><th>Input Parameter</th><th>Data Type</th><th colspan="3">Description</th><th>Valid and Default Values</th></tr>
<tr><td rowspan="11">Lock Direction (continued)</td><td rowspan="11">IMMEDIATE</td><td>Enumeration</td><td>Definition</td><td>Description</td><td rowspan="11">Valid = 0…4
Default = None
(Enumeration 1…4 are currently not allowed in Time Driven mode.)</td></tr>
<tr><td>3</td><td>Position Forward Only</td><td>Motion starts (that is, the Slave Coordinate System locks to the Master Axis) when the Master Axis crosses the Lock Position while it is moving in the Forward Direction. The Master Axis is only followed while it is moving in the Forward Direction.
Note that if the start position equals the Lock Position and this enumeration is selected, then motion will not start because the Lock Position will not be crossed.</td></tr>
<tr><td>4</td><td>Position Reverse Only</td><td>Motion starts when the Master Axis crosses the Lock Position while it is moving in the Reverse Direction. The Master Axis is only followed while it is moving in the Reverse Direction.
Note that if the start position equals the Lock Position and this enumeration is selected, then motion will not start because the Lock Position will not be crossed.</td></tr>
<tr><td colspan="3">For Time Driven Mode, the enumerations are as follows::</td></tr>
<tr><td>Enumeration</td><td>Definition</td><td>Description</td></tr>
<tr><td>0</td><td>None</td><td>Indicates that the Lock Position is not active.</td></tr>
<tr><td>1</td><td>Immediate Forward Only</td><td>The instruction will error with MDSC_LOCKDIR_CONFLICT (95).</td></tr>
<tr><td>2</td><td>Immediate Reverse Only</td><td>The instruction will error with MDSC_LOCKDIR_CONFLICT (95).</td></tr>
<tr><td>3</td><td>Position Forward Only</td><td>The instruction will error with MDSC_LOCKDIR_CONFLICT (95).</td></tr>
<tr><td>4</td><td>Position Reverse Only</td><td>The instruction will error with MDSC_LOCKDIR_CONFLICT (95).</td></tr>
<tr><td>Lock Position</td><td>IMMEDIATE REAL or TAG</td><td colspan="3">Lock Position in Master Driven Mode
The position on the Master Axis where a Slave Coordinate System should start after the move has been initiated on the Slave Coordinate System when executing in Master Driven Mode. This is an absolute position (plus or minus) on the Master Axis in Master Axis units. You can specify a Lock Position to delay the start of motion of a Slave Coordinate System after the motion instruction has been initiated on the Slave Coordinate System.
If an axis in the Slave Coordinate System is already moving and a coordinated move instruction (MCLM, or MCCM) with a Lock Position is activated on the Coordinate system, then you will receive an MDSC_LOCK_WHILE_MOVING error for the MCLM or MCCM instruction.
Because Merge is always performed immediately when an instruction is enabled, a merge instruction that starts at a nonzero velocity with both a Lock Position and a Merge enabled will receive an MDSC_LOCK_WHILE_MOVING error.
The Lock Direction determines the direction in which the Master Axis must be moving when it crosses the Lock Position before the Slave Coordinate System locks to the Master Axis.
Note that if there is an unwind value specified on the Master Axis, then the Lock Position must be between 0 and the unwind value (that is, the Lock Position cannot be more than one unwind.)
This parameter is only used in Master Driven Mode.</td><td>Default = 0.0</td></tr>
</table>

Table 66 - Input Parameters

Input Parameter	Data Type	Description	Valid and Default Values
Lock Position (continued	IMMEDIATE REAL or TAG	**Lock Position in Time Driven Mode** There is no Lock Position in Time Driven Mode for a coordinate system. An error will be generated if the Lock Direction is not NONE and the system is in Time Driven Mode for an MCLM or MCCM. This parameter is only used in Master Driven Mode. **Axis Lock Behavior** When the Master axis crosses the Master Lock position in the direction as specified by the motion instruction, the Slave Coordinate System becomes locked to the Master axis. The LockStatus bit is set at this time. The MCLM and MCCM instructions on the Slave Coordinate System in MDSC mode go IP as soon as they reach the head of the motion queue. The head of the queue is defined as the move right before the active move. For the Immediate Forward Only or Immediate Reverse Only Lock Directions, the Slave Coordinate System gets locked to the Master Axis immediately when the MCLM or MCCM instruction is executed (goes IP). For the Position Forward Only or Position Reverse Only Lock Directions, the slave gets locked to the Master Axis when the Master Axis crosses the Master Lock Position in the direction as specified by the motion instruction. In either case, the LockStatus bit is set when the lock occurs. Because there is no bi-directional behavior defined, once locked, the Slave Coordinate System follows the Master only in the specified direction. If the Master reverses direction, then the Slave stops following the Master. Note that the LockStatus bit remains set until the Master decelerates to zero. It is cleared at the point of reversal of the Master axis. The Slave does not follow the Master while the Master travels in the reverse direction. If the Master axis changes directions again, then the axis LockStatus bit is set again when the Slave Coordinate System crosses the original reversal point, at which time the slave resumes following the Master Axis. See the following illustration for clarification:	Default = 0.0

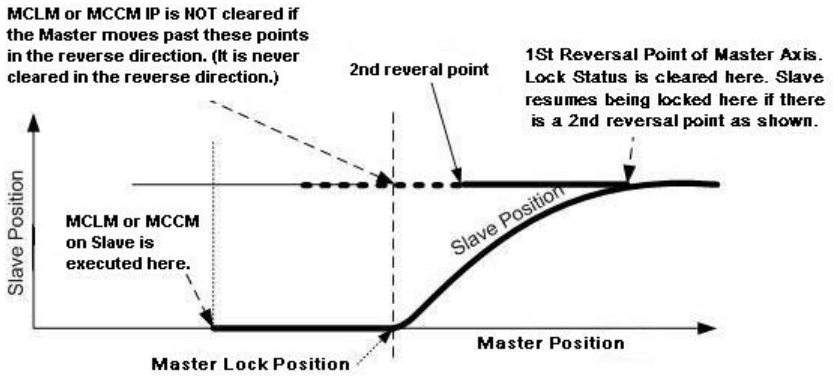
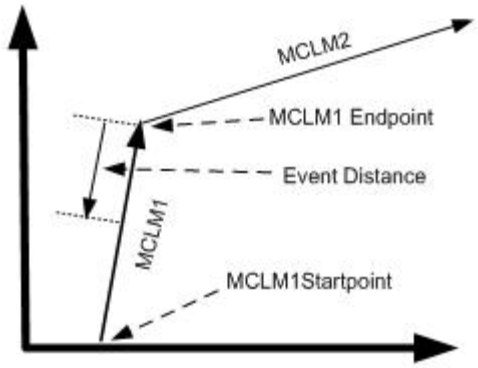

MCLM or MCCM IP is NOT cleared if the Master moves past these points in the reverse direction. (It is never cleared in the reverse direction.)
2nd reveral point
1St Reversal Point of Master Axis. Lock Status is cleared here. Slave resumes being locked here if there is a 2nd reversal point as shown.
Slave Position
MCLM or MCCM on Slave is executed here.
Slave Position
Master Position
Master Lock Position

Table 66 - Input Parameters

Input Parameter	Data Type	Description	Valid and Default Values
Lock Position (continued	IMMEDIATE REAL or TAG	On the Slave Coordinate System the following restrictions apply: • If a new instruction succeeds the active motion instruction but it is in the opposite direction of its current direction, then the error MDSC_LOCK_DIR_MASTER_DIR_MISMATCH is generated on the new motion instruction when it goes IP. The same is true if the new instruction is started via a merge operation. • A new instruction merged to an active instruction on the Slave Coordinate System must use the Immediate Forward Only or Immediate Reverse Only Lock Direction. If the new instruction uses the Position Forward Only or Position Reverse Only Lock Direction, the error MDSC_LOCKDIR_CONFLICT is generated on the new instruction. • A Lock Position may be used on an instruction that is merging a paused or dwelling motion instruction. On the Master Axis, no special restrictions apply. Note that if an instruction with the merge enabled is enqueued, then the whole queue is flushed and the active move is terminated. Note that if Master Axis filtering is enabled on the master axis, then the lock position for the Slave Coordinate System is delayed by the filter; the amount of delay is dependent on the filter bandwidth.	Default = 0.0
Command Tolerance	IMMEDIATE REAL or TAG	The position on a coordinated move where blending should start. When Termination Type 6 is used, the Command Tolerance on the instruction faceplate is used instead of the value for the Command Tolerance that is configured in the Coordinate System.	Default = 0.0
Event Distance	ARRAY or 0 (The array must be a minimum size of 4. If the array is greater than 4, only the first four locations specified are used.)	The position(s) on a move measured from the end of the move. This is an array of input values that specifies the incremental distances along the move on the SIave Coordinate System. Each member of the array is measured as follows: Distances are measured starting from the end of the move towards the beginning of the move as shown in the following Figure.  • For a linear coordinated move instructions (MCLM), the parameter value in the Event Distance can be represented as a vector starting at the move's end point and pointing towards the beginning of the move. • For a circular coordinated move (MCCM), the parameter value in the Event Distance is an incremental distance measured along the circular arc (that is, arc length) starting at the move's endpoint and moving towards the beginning of the circular arc. If the value in the Event Distance array is 0.0, then it is the time or distance for the whole move. The values entered in the Event Distance array are the same for both Time Driven and Master Driven Mode. Only the returned values in the Calculated Data array are different depending on the programmed mode of the Slave Coordinated System. When Event Distance is specified as a negative number, then the Event Distance calculation is skipped and a -1 is returned in the Calculated Data array for the specified Event Distance parameter. There is no limit on the dimension of either the Event Distance or Calculated Data arrays. However, only a maximum of 4 elements (the specified value and the next 3) of the Event Distance array will be processed. Note that special consideration for the rare case of an overshoot when an MCD or MCCD is done close to the moves endpoint. For this case, the Calculated Data will include the overshoot when the Event Distance is 0, since the master will have to traverse this amount for the move to finish. For other Event Distances, the overshoot will not be included.	Default = 0 (no Event Distance array)

The following table describes the output parameter.

Table 67 - Output Parameter

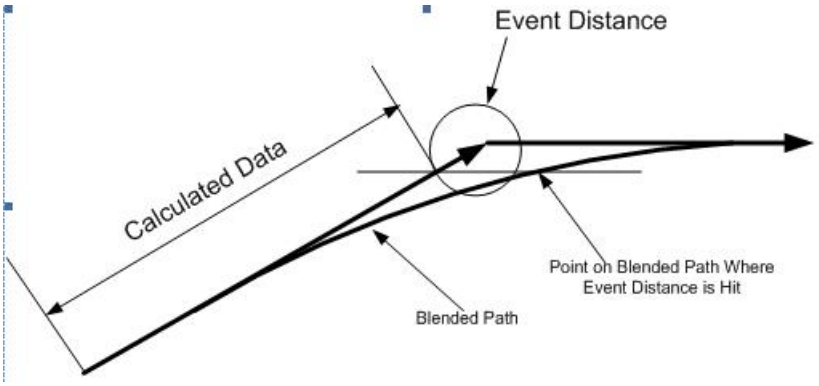
<table>
<tr><th>Output Parameter</th><th>Data Type</th><th colspan="2">Description</th><th>Valid and Default Values</th></tr>
<tr><td>Calculated Data</td><td>REAL
ARRAY or 0</td><td colspan="2">This is the Master Distance(s) (or time) needed for the Slave Coordinated System to travel from the beginning of the move to the Event Distance point.
The returned Calculated Data value is dependent on:
• the instruction type, (that is, MCLM or MCCM for coordinated motion).
• the mode of the Slave Coordinate System (that is, Time Driven or Master Driven).
• if superimposed motion is active, the Calculated Data does not include any of the superimposed motion.
To understand the Calculated Data concept, it's important to understand that the Motion Start Point (MSP) for a coordinated move is defined as the last time that:
• a TT0/TT1 was programmed, or completed or
• the queue was empty, or
• a merge occurred.
If there was a dwell programmed in the queue, then the calculated data includes the time of the dwell. Note that the MSP could have occurred several moves prior to the move in which the Event Distance was specified.
The returned Calculated Date value is outlined in the following table.</td><td>Default = 0 (no Calculated Data array) or a REAL array tag</td></tr>
<tr><td></td><td></td><td>Mode</td><td>Returned Calculated Data Parameter</td><td></td></tr>
<tr><td></td><td></td><td>Master Driven</td><td>The returned Calculated Data parameter is the incremental delta Master position that is needed to make the Slave Coordinate System move from the point at which Slave Coordinate System is locked to the Master and starts moving along the programmed path to the point where distance to go is less than the specified Event Distance. (See Example 3 below. In example 3, the MSP for all event distances is point P0.)</td><td></td></tr>
<tr><td colspan="5"> 
</td></tr>
<tr><td>Calculated Data (continued)</td><td>REAL
ARRAY or 0</td><td>Mode</td><td>Returned Calculated Data Parameter</td><td>Default = 0 (no Calculated Data array) or a REAL array tag</td></tr>
<tr><td></td><td></td><td>Master Driven</td><td>• For Blended moves (that is, Termination Type =Command Tolerance or No Decel).
The incremental Master Axis distance needed for the programmed move, in the Slave Coordinate System, to travel from the beginning of the move to the Blend Point. Note that this is where the PC bit of the instruction is set.
• For all other termination types (that is, non-blended moves)
The incremental Master Axis distance needed for programmed move, in the Slave Coordinate System, to travel from the beginning of the move to the programmed endpoint. Note that this is where the PC bit of instruction is set on the instruction moving the slave.
• Another way to represent the Event Distance and the corresponding Calculated Data is on a Velocity versus Time plot as is shown in the following figure: Note that the first plot below is for non-blended moves (TT0/1), the second is for blended (TT2, 3, 6).</td><td></td></tr>
</table>

Table 67 - Output Parameter

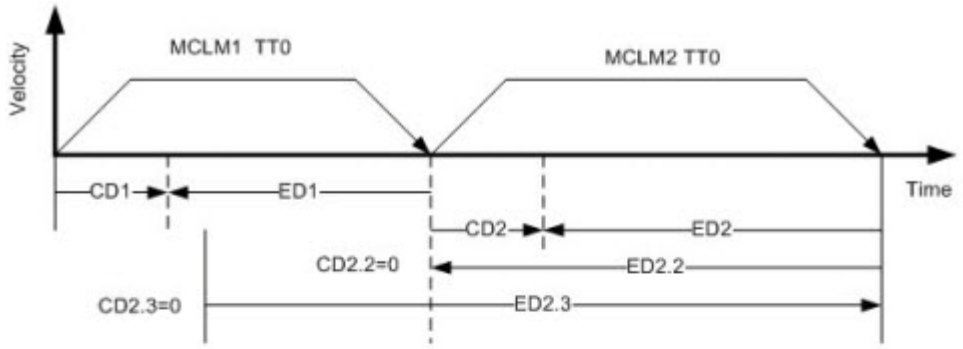
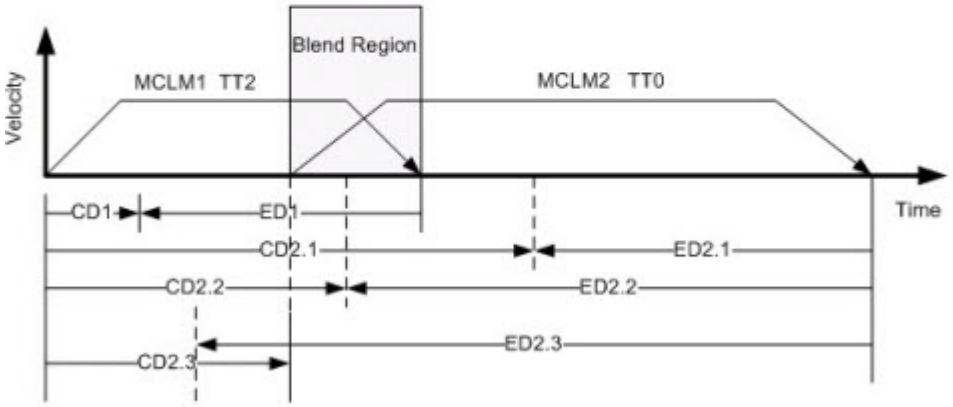
Output Parameter	Data Type	Description	Valid and Default Values
		  CD = Calculated Data ED = Event Distance TT = Termination Type CD2.1 = Calculated Data 1 for Move 2 ED2.3 = Event Distance 3 for Move 2 CD2.1 is how far the master has to move when the slave reaches position ED2.1 **Note**: As shown above, the Event Distance is > Move length and is therefore internally set equal to the move length. The Calculated Data for the move length is therefore returned. No error is forced.	

Table 67 - Output Parameter

<table>
<tr><th>Output Parameter</th><th>Data Type</th><th colspan="2">Description</th><th>Valid and Default Values</th></tr>
<tr><td rowspan="3">Calculated Data (continued)</td><td rowspan="3">REAL
ARRAY or 0</td><td>Mode</td><td>Returned Calculated Data Parameter</td><td rowspan="3">Default = 0 (no Calculated Data array) or a REAL array tag</td></tr>
<tr><td>Time Driven</td><td>The returned data in the Calculated Data parameter is the total time in seconds that is needed to make the Slave Coordinate System move from the move's start point to a point where distance to go is less than the specified Event Distance. If the specified data in the Event Distance is array element is 0.0, then the time it takes the entire move to complete is returned.</td></tr>
<tr><td colspan="2">The Logix Designer application Motion Planner processes and calculates output data and places the result in the Calculated Data array as supplied in the instruction. The number of calculated array elements stored in the Calculated Data array is based on the follow conditions:
• The number of elements in the Event Distance array.
• For each of the first 4 elements Event Data array, one element is computed and placed in the Calculated Data array.
• The fifth element and beyond of the Event Distance array are ignored. Existing values in the Calculated Data array are overlaid when the Event Distance array is processed.
A -1 is returned in the Calculated Data array for each negative value in the Event Distance array. No Event Distance calculation is made for these array elements.
You can change the Event Distance array elements dynamically in the program. However, if the Event Distance is changed after the instruction has been initiated (that is, the IP bit has been set), then the change is ignored.
An error is generated if the size of the Calculated Data array is smaller than the Event Distance array.
If the Event Distance is greater than the move length internally (vector length for MCLM, arc length for MCCM), it is forced to equal the move length.
If a MCD or MCCD is executed (indicated by status bit going IP), the CalculatedDataAvailable (CDA) bit is cleared. The Calculated Data for the move is recomputed by using the new dynamics parameters. Only those items of the Calculated Data array whose Event Distance has not been reached yet are recomputed; other items are left as they are. Consequently, all Calculated Data array items contain valid information after the move is completed. The CDA bit is set again when computations are complete, The Calculated Data that is recomputed is measured from the original MSP to the Event Distance point by using the new dynamics parameters as changed by the MCD or MCCD instruction, not from the point of the MCD or MCCD. Note that if the MCD changes the speed to 0, the Event Distance is not recomputed; the CDA bit is cleared and stay cleared. The Event Distance, however, is recomputed if a second MCD or MCCD is issued to restart the motion. The recomputed Calculated Data includes the duration of the stopped motion.
If the Event Distance is set to 0, the Calculated Data is set equal to the position that equals the length of the move. This can be one or two coarse update periods before the PC bit is set because of an internal delay. The end position is typically achieved in the middle of a coarse update period that adds up to one additional coarse update period to the delay. Therefore, if the master is moved a distance equal to the Calculated Data, you must wait up to 2 iterations more for the PC bit of the slave move to be set.
Note that there is a special consideration for the rare case of an overshoot when an MCD or MCCD is done close to the moves endpoint. For this case, when the Event Distance is 0, the returned Calculated Data includes the overshoot distance traveled, because the master has to traverse this amount for the move to finish. For non-zero Event Distances, the overshoot distance is not included.
A status bit (CalculatedDataAvailable) in the existing motion instruction status word has been defined to indicate that all of the requested data for the specified Event Distance array elements has been returned in the corresponding Calculated Data array elements. Only one status bit is used to indicate all Calculated Data is available.</td></tr>
</table>

Table 67 - Output Parameter

Output Parameter	Data Type	Description	Valid and Default Values
Calculated Data (continued)	REAL ARRAY or 0	Once set, this bit can later be cleared based on a number of different conditions including, but not limited to, an MAS, MCS being executed. Note that Calculated Data is only set once in the instruction queue or planning process. It is not updated as the move occurs to reflect distance to go. It is updated for a change dynamics, however. For coordinated moves, the CalculatedDataAvailable status bit is set when the Calculated Data is available. In general, for a blending termination type (TT2, 3, 6) or follow contour termination type (TT4, 5), you do not see CalculatedDataAvailable for move N until move N+1 is put in the queue. For a non-blended termination type (TT0, 1), the CalculatedDataAvailable is seen right after the move is put into the queue. You do not see the CalculatedDataAvailable bit if a move sequence is terminated by a blending or follow contour termination type. That is, you must terminate a blending sequence by a TT0 or TT1. The TT0 or TT1 has to be in the motion sequence, but does not have to be in the queue together with a blending sequence. The move with a TT0 or TT1 can be placed in the queue when space becomes available after the last blended move. The CalculatedDataAvailable bit is not set for any move that Event Distance is not specified, that is, where the Event Distance parameter in the instruction is zero. The default value for versions when bringing old systems forward (earlier than v20) is 0, signifying that there is no Event Distance array. **Example 1** Event Distance array = [11, 22, -5, 23, 44] Calculated Data array = [f(11), f(22), -1 ,f(23)] Where f is the calculated data function. **Note:** • The 44 is ignored because it is the fifth element in the Event Distance array. Nothing is returned in the corresponding 5th array element of Calculated Data array. • A -1 is returned in the third element of the Calculated Data array because the corresponding Event Data Array element is negative. **Example 2** Assume that the master axis is at a position of 2.0. The slave is programmed to an incremental value of 15.0 with a Master Lock Position at 8.0. The Event Distance is set to 0.0, which means that we want the total Master Distance (X in the diagram below) needed for the slave to move 15.0 units starting when the Master is locked at a position at 8.0. The incremental value of X is returned in the Calculated Data parameter.	Default = 0 (no Calculated Data array) or a REAL array tag

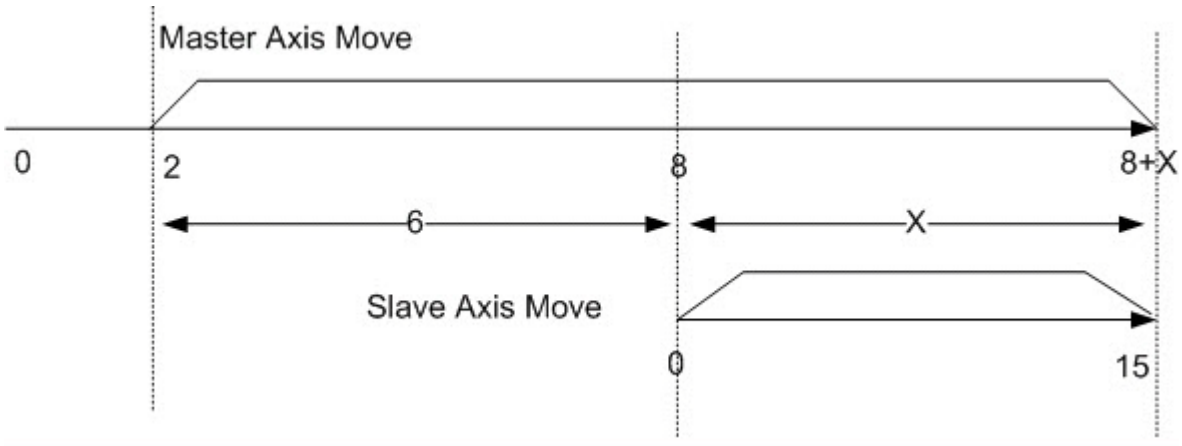
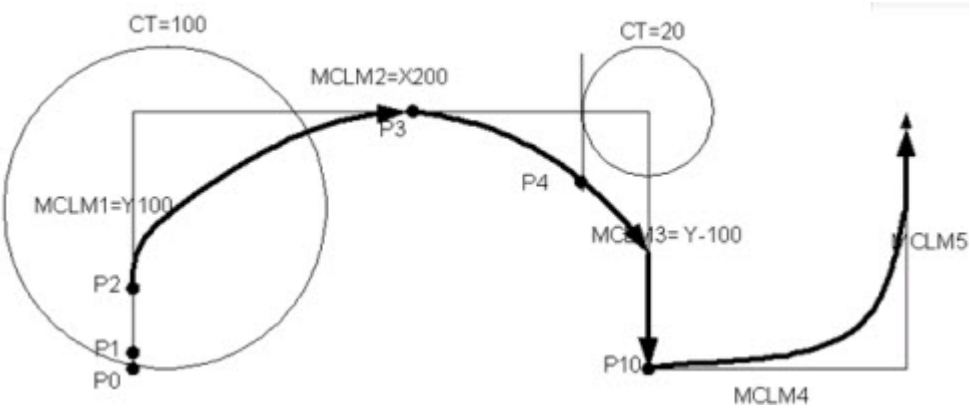


Table 67 - Output Parameter

Output Parameter	Data Type	Description	Valid and Default Values
Calculated Data (continued)	REAL ARRAY or 0	**Example 3** The following example illustrates using Event Distance and Calculated Data. Note that the MSP for all event distances is point P0. The MSP is where the Slave. is locked to the Master and starts moving along the programmed path.	Default = 0 (no Calculated Data array) or a REAL array tag
			
Calculated Data (continued)	REAL ARRAY or 0	5 Move Segments are specified Event Distance = ED Command Tolerance = CT • MCLM1 Y100; TT2 ED=50 CT=100 • MCLM2 X200; TT2 ED=100 CT=20 • MCLM3 Y-100; TT1 ED=100 CT=20 • MCLM4 X200; TT2 ED=100 CT=20 • MCLM5 Y100; TT2 ED=100 CT=20 The calculated data for MCLM1 is returned when MCLM2 is added to the queue and planned. This is at point P1 above. Master Distance is Measured from point P0. The calculated data for MCLM2 is returned when MCLM3 is added to the queue and planned. This is at Point P2 above. Master Distance is Measured from point P0. The calculated data for MCLM3 is returned when MCLM3 is added to the queue and planned. This is at point P3 above. Master Distance is Measured from point P0. The calculated data for MCLM4 is returned when MCLM5 is added to the queue and planned. This is at point P10 above. Master Distance is Measured from point P10. The calculated data for MCLM5 is never returned because MCLM5 is terminated with TT2 and it is the last move in the queue. Use TT0 or TT1 instead. All Calculated Data is the master distance or time from the last MSP point. That is, it is where the slave is at rest, which is point P0 and point P10 above.	Default = 0 (no Calculated Data array) or a REAL array tag

Speed, Acceleration, Deceleration, and Jerk Enumerations for Coordinated Motion

Speed Enumerations

Common enumerations are used for the speed parameter of all motion instructions. Some instructions accept only limited subset of the speed enumerations. Checks for valid unit combinations are done at instruction execution time. Some enumerations that are in the following table are not used now but are reserved for future enhancements. Additional tables are given below that further clarify which combinations are accepted in MDSC mode and which are accepted in Time Driven Mode.

Table 68 - Speed Enumeration

Enumeration	Definition	Mode	Compatibility	Note
0	Units per sec	Time Driven	Existing Enumeration	
1	% Maximum		Existing Enumeration	
2	Reserved			
3	Reserved		New Enumeration	Reserved for Time based programming
4	Units per MasterUnit	MDSC	New Enumeration	
5	Reserved			
6	Reserved			
7	Master Units		New Enumeration	Reserved for Time based p7ogramming

These rules for Speed must be followed to determine allowable Time and MDSC Driven Mode:

- When Speed is in either units/sec, %max, or seconds, then the instruction is considered to be in Time Driven Mode, regardless of the selection of units for acceleration, deceleration, or jerk.
- When Speed is in either Master Units or in Units/MasterUnit, then the instruction is considered to be in Master Driven Mode, regardless of the selection of units for acceleration, deceleration, or jerk.
- Speed, Acceleration, Deceleration, and Jerk must always be programmed in the same mode (Time Driven or Master Driven) or you get a runtime error.
- When speed is specified in time unit seconds, the specified time is the total time of the move, including acceleration and deceleration time.

When speed is specified in Master distance units, the specified distance is the total master distance of the move, including acceleration and deceleration distance of the Master Axis.

Acceleration and Deceleration Enumerations

The following enumerations are defined for Acceleration and Deceleration Unit parameters for motion instructions.

Table 69 - Acceleration and Deceleration Enumeration

Enumeration	Definition	Mode	Compatibility	Note
0	Units per sec^2	Time Driven	Existing Enumeration	
1	% Maximum		Existing Enumeration	
2	Reserved			
3	Reserved			Reserved for Time based programming
4	Units per MasterUnit^2	MDSC	New Enumeration	
5	Reserved			
6	Reserved			
7	Master Units			Reserved for Time based programming

The following table shows acceptable combinations of Speed, Acceleration, and Deceleration units.

Table 70 - Acceleration

		Acceleration and Deceleration Units				
		Units per sec^2 (Time Driven Mode Units)	**% Maximum (Time Driven Mode Units)**	**Seconds (Time Driven Mode Units)**	**Units per MasterUnit^2 (Master Driven Mode Units)**	**Master Units (Master Driven Mode Units)**
Speed Units	**Units per sec (Time Driven Mode Units)**	Existing Enumeration	Existing Enumeration	Not Implemented	Not allowed - Time and Master Driven Units can not be combined.	
	% Maximum (Time Driven Mode Units)	Existing Enumeration	Existing Enumeration	Not Implemented		
	Seconds (Time Driven Mode Units)	Not Implemented	Not Implemented	New Enumeration		
	Units per MasterUnits (Master Driven Mode Units)	Not allowed - Time and Master Driven Units cannot be combined.			New Enumeration	Not Implemented
	Master Units (Master Driven Mode Units)				Not Implemented	New Enumeration

These rules for Acceleration and Deceleration must be followed to determine allowable Time and Master Driven Mode:

- Speed, Acceleration, Deceleration, and Jerk must always be programmed in the same mode or you get an error.
- If Speed units are seconds, then acceleration, deceleration, and jerk units must be seconds too.
- If Speed units are Master units, then acceleration, deceleration, and jerk units must be Master units too.
- All unsupported unit combinations result in an error at runtime when the instruction is executed.

Jerk Enumerations

The following enumerations are defined for time driven and MDSC driven Jerk units.

Table 71 - Jerk Enumeration

Enumeration	Description	Mode	Compatibility	Notes
0	Units per sec^3	Time	Existing Enumeration	
1	% Maximum		Existing Enumeration	
2	% of Time		Existing Enumeration	
3	Reserved			Reserved for Time based programming
4	Units per MasterUnit^3	MDSC	New Enumeration	
5	Reserved			
6	% of Time-Master Driven		New Enumeration	
7	Reserved			Reserved for Time based programming

Acceptable combinations of Accel and Decel Units are based on the programmed Speed Units in the instruction as is shown in the table below. This table is used to clarify the differences in the following four tables.

Speed Units	Accel Units vs Jerk Units Defined in Table:
Units per Sec	Table 72
Units per Master Units	Table 73
Seconds	Table 74
Master Units	Table 75

The following table shows acceptable combinations of Acceleration Units and Jerk Units when Speed Units are Units per Sec.

Table 72 - Speed Units in Units per Sec

		Acceleration Units (Speed in Units per Second)				
		Units per sec[2] (Time Driven Mode Units)	**% Maximum (Time Driven Mode Units)**	**Seconds (Time Driven Mode Units)**	**Units per MasterUnit[2] (Master Driven Mode Units)**	**Master Units (Master Driven Mode Units)**
Jerk Units	**Units per sec[3] (Time Driven Mode Units)**	Existing Enumeration.	Existing Enumeration.	Not Implemented	Incompatible combinations of Time and Master Driven Mode. A runtime error occurs.	
	% Maximum (Time Driven Mode Units)	Existing Enumeration.	Existing Enumeration.	Not Implemented		
	% of Time (Time Driven Mode Units)	Existing Enumeration.	Existing Enumeration.	Not Implemented		
	Seconds (Time Driven Mode Units)	Not Implemented	Not Implemented	Not Implemented		
	Units per MasterUnits[3] (Master Driven Mode Units)	Incompatible combinations of Time and Master Driven Mode. An error occurs when you verify the routine.			Incompatible combinations of Time and Master Driven Mode. An error occurs when you verify the routine.	
	% of Time-Master Driven (Master Driven Mode Units)					
	Master Units (Master Driven Mode Units)					

The following table shows acceptable combinations of Acceleration Units and Jerk Units when Speed Units are Units per Master Unit.

Table 73 - Speed Units in Units per Master Units

		Acceleration (Speed in Units per Master Unit)				
		Units per sec[2] (Time Driven Mode Units)	**% Maximum (Time Driven Mode Units)**	**Seconds (Time Driven Mode Units)**	**Units per MasterUnit[2] (Master Driven Mode Units)**	**Master Units (Master Driven Mode Units)**
Jerk Units	**Units per sec[3] (Time Driven Mode Units)**	Incompatible combinations of Time and Master Driven Mode. An error occurs when you verify the routine.			Incompatible combinations of Time and Master Driven Mode. An error occurs when you verify the routine.	
	% Maximum (Time Driven Mode Units)					
	% of Time (Time Driven Mode Units)					
	Seconds (Time Driven Mode Units)					
	Units per MasterUnits[3] (Master Driven Mode Units)	Incompatible combinations of Time and Master Driven Mode. An error occurs when you verify the routine.			New Enumeration.	Not Implemented
	% of Time-Master Driven (Master Driven Mode Units)				New Enumeration.	Not Implemented
	Master Units (Master Driven Mode Units)				Not Implemented	Not Implemented

The following table shows acceptable combinations of Acceleration Units and Jerk Units when Speed Units are in Seconds.

Table 74 - Speed Units in Seconds

		Acceleration (Speed in Seconds)				
		Units per sec^2 (Time Driven Mode Units)	**% Maximum (Time Driven Mode Units)**	**Seconds (Time Driven Mode Units)**	**Units per MasterUnit^2 (Master Driven Mode Units)**	**Master Units (Master Driven Mode Units)**
Jerk Units	**Units per sec^3 (Time Driven Mode Units)**	Not Implemented	Not Implemented	Not Implemented	Incompatible combinations of Time and Master Driven Mode. An error occurs when you verify the routine.	
	% Maximum (Time Driven Mode Units)	Not Implemented	Not Implemented	Not Implemented		
	% of Time (Time Driven Mode Units)	Not Implemented	Not Implemented	New Enumeration.		
	Seconds (Time Driven Mode Units)	Not Implemented	Not Implemented	New Enumeration.		
	Units per MasterUnits^3 (Master Driven Mode Units)	Incompatible combinations of Time and Master Driven Mode. An error occurs when you verify the routine.			Incompatible combinations of Time and Master Driven Mode. An error occurs when you verify the routine.	
	% of Time-Master Driven (Master Driven Mode Units)					
	Master Units (Master Driven Mode Units)					

The following table shows acceptable combinations of Acceleration Units and Jerk Units when Speed is in Master Units.

Table 75 - Speed Units in Master Units

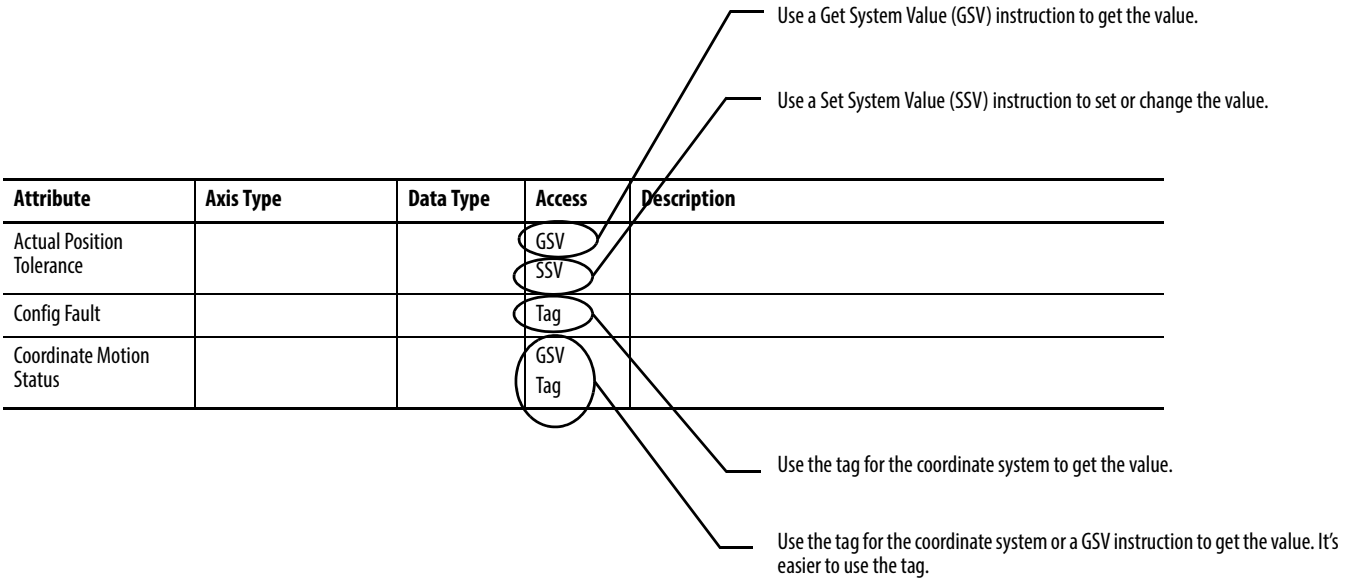
		Acceleration (Speed in MasterUnits)				
		Units per sec^2 (Time Driven Mode Units)	**% Maximum (Time Driven Mode Units)**	**Seconds (Time Driven Mode Units)**	**Units per MasterUnit^2 (Master Driven Mode Units)**	**Master Units (Master Driven Mode Units)**
Jerk Units	**Units per sec^3 (Time Driven Mode Units)**	Incompatible combinations of Time and Master Driven Mode. An error occurs when you verify the routine.			Incompatible combinations of Time and Master Driven Mode. An error occurs when you verify the routine.	
	% Maximum (Time Driven Mode Units)					
	% of Time (Time Driven Mode Units)					
	Seconds (Time Driven Mode Units)					
	Units per MasterUnits^3 (Master Driven Mode Units)	Incompatible combinations of Time and Master Driven Mode. An error occurs when you verify the routine.			Not Implemented	Not Implemented
	% of Time-Master Driven (Master Driven Mode Units)				Not Implemented	New Enumeration.
	Master Units (Master Driven Mode Units)				Not Implemented	New Enumeration.

Coordinate System Attributes

Use this appendix for information about the attributes used in a coordinate system.

How to Access Attributes

The Access column shows how you can access the attribute.


Use a Get System Value (GSV) instruction to get the value.
Use a Set System Value (SSV) instruction to set or change the value.
Use the tag for the coordinate system to get the value.
Use the tag for the coordinate system or a GSV instruction to get the value. It's easier to use the tag.

Attribute	Axis Type	Data Type	Access	Description
Actual Position Tolerance			GSV SSV	
Config Fault			Tag	
Coordinate Motion Status			GSV Tag	

Coordinate System Attributes

Attribute	Data Type	Access	Description
Accel Status	BOOL	Tag	Use the Accel Status bit to determine if the coordinated (vectored) motion is currently being commanded to accelerate. The acceleration bit is set when a coordinated move is in the accelerating phase due to the current coordinated move. It is cleared when the coordinated move has been stopped or the coordinated move is in the decelerating phase.
Actual Pos Tolerance Status	BOOL	Tag	Use the Actual Pos Tolerance Status bit to determine when a coordinate move is within the Actual Position Tolerance. The Actual Position Tolerance Status bit is set for AT term type only. The bit is set when interpolation is complete and the actual distance to programmed endpoint is less than the configured AT value. The bit remains set after an instruction completes. The bit is reset if either a new instruction is started or the axis moves such that the actual distance to programmed endpoint is greater than the configured AT value
Actual Position	REAL[8]	Tag	Array of actual position of each axis associated to this motion coordinate system in Coordinate Units.

<table>
<tr><th>Attribute</th><th>Data Type</th><th>Access</th><th>Description</th></tr>
<tr><td>Actual Position Tolerance</td><td>REAL</td><td>GSV
SSV</td><td>Coordination Units
The Actual Position Tolerance attribute value is a distance unit used when instructions (for example, MCLM and MCCM) specify a Termination Type of Actual Position.</td></tr>
<tr><td>Axes Configuration Faulted</td><td>DINT</td><td>GSV
Tag</td><td>Shows which axes in this coordinate system have a configuration fault.<table><tr><th>If this bit is on</th><th>Then this axis has a configuration fault</th></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td></tr><tr><td>2</td><td>2</td></tr></table></td></tr>
<tr><td>Axes Inhibited Status</td><td>DINT</td><td>GSV
Tag</td><td>Shows which axes in this coordinate system are inhibited.<table><tr><th>If this bit is on</th><th>Then this axis is inhibited</th></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td></tr><tr><td>2</td><td>2</td></tr></table></td></tr>
<tr><td>Axes Servo On Status</td><td>DINT</td><td>GSV
Tag</td><td>Shows which axes in this coordinate system are on (via MSO).<table><tr><th>If this bit is on</th><th>Then this axis is on</th></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td></tr><tr><td>2</td><td>2</td></tr></table></td></tr>
<tr><td>Axes Shutdown Status</td><td>DINT</td><td>GSV
Tag</td><td>Shows which axes in this coordinate system are shutdown.<table><tr><th>If this bit is on</th><th>Then this axis is shutdown</th></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td></tr><tr><td>2</td><td>2</td></tr></table></td></tr>
<tr><td>Axis Fault</td><td>DINT</td><td>GSV
Tag</td><td>The Axis Fault Bits attribute is a roll-up of all of the axes associated to this motion coordinate system. A bit being set indicates that one of the associated axes has that fault.<table><tr><th>Type</th><th>Bit</th></tr><tr><td>Physical Axis Fault</td><td>0</td></tr><tr><td>Module Fault</td><td>1</td></tr><tr><td>Config Fault</td><td>2</td></tr></table></td></tr>
<tr><td>Axis Inhibit Status</td><td>BOOL</td><td>Tag</td><td>If this bit is:
• ON — An axis in the coordinate system is inhibited.
• OFF — None of the axis in the coordinate system are inhibited.</td></tr>
<tr><td>Command Pos Tolerance Status</td><td>BOOL</td><td>Tag</td><td>Use the Command Position Tolerance Status bit to determine when a coordinate move is within the Command Position Tolerance.
The Command Position Tolerance Status bit is set for all term types whenever the distance to programmed endpoint is less than the configured CT value. The bit remains set after an instruction completes. The bit is reset when a new instruction is started.</td></tr>
<tr><td>Command Position Tolerance</td><td>REAL</td><td>GSV
SSV</td><td>Coordination Units
The Command Position Tolerance attribute value is a distance unit used when instructions (for example, MCLM and MCCM) specify a Termination Type of Command Position.</td></tr>
</table>

<table>
<tr><th>Attribute</th><th>Data Type</th><th>Access</th><th>Description</th></tr>
<tr><td>Config Fault</td><td>BOOL</td><td>Tag</td><td>The Configuration Fault bit is set when an update operation targeting an axis configuration attribute of an associated motion module has failed. Specific information concerning the Configuration Fault can be found in the Attribute Error Code and Attribute Error ID attributes associated with the motion module.</td></tr>
<tr><td>Coordinate Motion Status</td><td>DINT</td><td>GSV
Tag</td><td>Lets you access the motion status bits for the coordinate system in one 32-bit word.<table><tr><th>Status</th><th>Bit</th></tr><tr><td>Accel Status</td><td>0</td></tr><tr><td>Decel Status</td><td>1</td></tr><tr><td>Actual Pos Tolerance Status</td><td>2</td></tr><tr><td>Command Pos Tolerance Status</td><td>3</td></tr><tr><td>Stopping Status</td><td>4</td></tr><tr><td>Reserved</td><td>5</td></tr><tr><td>Move Status</td><td>6</td></tr><tr><td>Transition Status</td><td>7</td></tr><tr><td>Move Pending Status</td><td>8</td></tr><tr><td>Move Pending Queue Full Status</td><td>9</td></tr></table></td></tr>
<tr><td>Coordinate System Auto Tag Update</td><td>SINT</td><td>GSV
SSV</td><td>The Coordinate System Auto Tag Update attribute configures whether the Actual Position attribute is automatically updated each motion task scan. This is similar to, but separate from, the Motion Group's "Auto Tag Update" attribute.
0 – auto update disabled
1 – auto update enabled (default)</td></tr>
<tr><td>Coordinate System Status</td><td>DINT</td><td>GSV
Tag</td><td>Lets you access the status bits for the coordinate system in one 32-bit word.<table><tr><th>Status</th><th>Bit</th></tr><tr><td>Shutdown Status</td><td>0</td></tr><tr><td>Ready Status</td><td>1</td></tr><tr><td>MotionStatus</td><td>2</td></tr><tr><td>Axis Inhibit Status</td><td>3</td></tr></table></td></tr>
<tr><td>Decel Status</td><td>BOOL</td><td>Tag</td><td>Use the Decel Status bit to determine if the coordinated (vectored) motion is currently being commanded to decelerate.
The deceleration bit is set when a coordinated move is in the decelerating phase due to the current coordinated move. It is cleared when the coordinated move has been stopped or the coordinated move is complete.</td></tr>
</table>

<table>
<tr><th>Attribute</th><th>Data Type</th><th>Access</th><th>Description</th></tr>
<tr><td>Dynamics Configuration Bits</td><td>DINT</td><td>GSV
SSV</td><td>Revision 16 improved how the controller handles changes to an S-Curve profile.
Do you want to return to revision 15 or earlier behavior for S-Curves?
• NO — Leave these bits ON (default).
• YES — Turn OFF one or more of these bits.
<table><tr><th>To turn off this change</th><th>Turn off this bit</th></tr><tr><td>Reduced S-Curve Stop Delay
This change applies to the Motion Coordinated Stop (MCS) instruction. It lets you use a higher deceleration jerk to stop an accelerating coordinate system more quickly.
The controller uses the deceleration jerk of the stopping instruction if it is more than the current acceleration jerk.</td><td>0</td></tr><tr><td>Reduced S-Curve Velocity Reversals
Before revision 16, you could cause a coordinate system to momentarily reverse direction if you decreased the deceleration jerk while the coordinate system was decelerating. This typically happened if you tried to restart a move with a lower deceleration rate while the coordinate system was stopping. This change prevents the coordinate system from reversing in those situations.</td><td>1</td></tr><tr><td>Reduced S-Curve Velocity Overshoots
You can cause a coordinate system to overshoot its programmed speed if you decrease the acceleration jerk while the coordinate system is accelerating. This change keeps to overshoot to no more than 50% of the programmed speed.</td><td>2</td></tr></table></td></tr>
<tr><td>Maximum Acceleration</td><td>REAL</td><td>GSV
SSV</td><td>Coordination Units / Sec^2
The Maximum Acceleration attribute value is used by motion instructions (for example, MCLM and MCCM), to determine the acceleration rate to apply to the coordinate system vector when the acceleration is specified as a percent of the Maximum.</td></tr>
<tr><td>Maximum Deceleration</td><td>REAL</td><td>GSV
SSV</td><td>Coordination Units / Sec^2
The Maximum Deceleration attribute value is used by motion instructions s (for example, MCLM and MCCM), to determine the deceleration rate to apply to the coordinate system vector when the deceleration is specified as a percent of the Maximum.</td></tr>
<tr><td>Maximum Pending Moves</td><td>DINT</td><td>GSV</td><td>The Maximum Pending Moves attribute is used to determine how many Move Pending queue slots are created as part of the Coordinate System's create service.
Limited to a queue of one.</td></tr>
<tr><td>Maximum Speed</td><td>REAL</td><td>GSV
SSV</td><td>Coordination Units / Sec
The value of the Maximum Speed attribute is used by various motion instructions (for example, MCLM, MCCM and so on) to determine the steady-state speed of the coordinate system vector when the speed is specified as a percent of the Maximum.</td></tr>
<tr><td>Module Fault</td><td>BOOL</td><td>Tag</td><td>The Module Fault bit attribute is set when a serious fault has occurred with the motion module associated with the selected axis. Usually a module fault affects all axes associated with the motion module. A module fault generally results in the shutdown of all associated axes. Reconfiguration of the motion module is required to recover from a module fault condition.</td></tr>
<tr><td>Modules Faulted</td><td>DINT</td><td>GSV
Tag</td><td>Shows which axes in this coordinate system have a module fault.
<table><tr><th>If this bit is on</th><th>Then this axis has a module fault</th></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td></tr><tr><td>2</td><td>2</td></tr></table></td></tr>
<tr><td>Motion Status</td><td>BOOL</td><td>Tag</td><td>The Motion Status bit attribute is set indicating that at least one Coordinate Motion instruction is active and the Coordinate System is connected to its associated axes.</td></tr>
<tr><td>Move Pending Queue Full Status</td><td>BOOL</td><td>Tag</td><td>The move pending queue full bit is set when there is no room in the instruction queue for the next coordinated move instruction. Once there is room in the queue, the bit is cleared.</td></tr>
</table>

<table>
<tr><th>Attribute</th><th>Data Type</th><th>Access</th><th>Description</th></tr>
<tr><td>Move Pending Status</td><td>BOOL</td><td>Tag</td><td>The move pending bit is set once a coordinated motion instruction is queued. Once the instruction has begun executing, the bit is cleared, provided no subsequent coordinated motion instructions have been queued in the meantime.
In the case of a single coordinated motion instruction, the status bit can fail to be detected in Logix Designer application because the transition from queued to executing is faster than the coarse update. The real value of the bit comes in the case of multiple instructions. As long as an instruction is in the instruction queue, the pending bit is set.
This provides the Logix Designer programmer a means of stream-lining the execution of multiple coordinated motion instructions. Ladder logic containing coordinated motion instructions can be made to execute faster when the programmer lets instructions be queued while a preceding instruction is executing. When the MovePendingStatus bit is clear, the next coordinated motion instruction can be executed (that is, setup in the queue).</td></tr>
<tr><td>Move Status</td><td>BOOL</td><td>Tag</td><td>The move bit is set when coordinated motion is generating motion for any associated axes. Once coordinated motion is no longer being commanded, the move bit is cleared.</td></tr>
<tr><td>Move Transition Status</td><td>BOOL</td><td>Tag</td><td>The move transition bit is set once the blend point between two successive coordinated moves has been reach. The bit remains set while the blend of the two moves into one is in process. Once the blend is complete, the move transition bit is cleared.</td></tr>
<tr><td>Physical Axes Faulted</td><td>DINT</td><td>GSV
Tag</td><td>Shows which axes in this coordinate system have a servo axis fault.
<table>
<tr><th>If this bit is on</th><th>Then this axis has a servo axis fault</th></tr>
<tr><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
<tr><td>2</td><td>2</td></tr>
</table></td></tr>
<tr><td>Physical Axis Fault</td><td>BOOL</td><td>Tag</td><td>If the Physical Axis Fault bit is set, it indicates that there is one or more fault conditions that have been reported by the physical axis. The specific fault conditions can then be determined through access to the fault attributes of the associated physical axis.</td></tr>
<tr><td>Ready Status</td><td>BOOL</td><td>Tag</td><td>The Ready bit is set when all associated axes are enabled. It is cleared after an MCSD, MGSD or a fault on any of the associated axes.</td></tr>
<tr><td>Shutdown Status</td><td>BOOL</td><td>Tag</td><td>The Coordinate System bit is set after an MCSD or MGSD is executed and all associated axes have stopped. An MCSR or a MGSR resets the coordinate system and clear the bit. Coordinated moves cannot be initiated while this bit is set.</td></tr>
<tr><td>Stopping Status</td><td>BOOL</td><td>Tag</td><td>The stopping bit is set when an MCS instruction is executed. The bit remains set until all coordinated motion is stopped. The bit is cleared when all coordinated motion has stopped.</td></tr>
<tr><td>Transform Source Status</td><td>BOOL</td><td>Tag</td><td>If the bit is:
• ON — The coordinate system is the source of an active transform.
• OFF — The coordinate system isn't the source of an active transform.</td></tr>
<tr><td>Transform Target Status</td><td>BOOL</td><td>Tag</td><td>If the bit is:
• ON — The coordinate system is the target of an active transform.
• OFF — The coordinate system is not the target of an active transform.</td></tr>
</table>

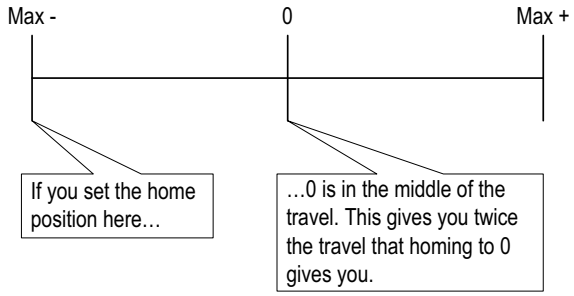
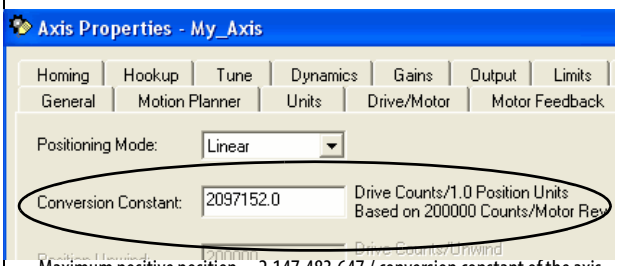
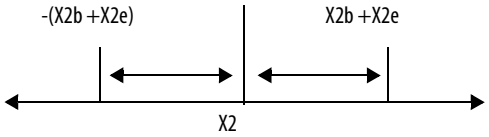
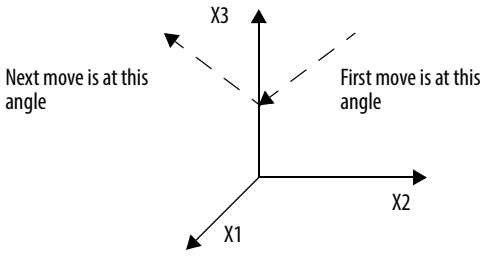
Notes:

Error Codes (ERR) for Coordinate Motion Instructions

Error	Corrective Action or Cause	Notes
1	Reserved Error Code 1.	Reserved for future use.
2	Reserved Error Code 2.	Reserved for future use.
3	Look for another instance of this type of instruction. See if its EN bit is on but its DN and ER bits are off (enabled but not done or errored). Wait until its DN or ER bit turns on.	Execution Collision You cannot execute an instruction if the same type of instruction is enabled but not done or errored.
4	Open the servo loop before you execute this instruction.	Servo On State Error
5	Close the servo loop before you execute this instruction.	Servo Off State Error For a motion coordinated instruction, look at the extended error code (EXERR). It identifies which axis caused the error. **Example**: If EXERR is zero, check the axis for dimension zero.
6	Disable the axis drive.	Drive On State Error
7	Execute a Motion Axis Shutdown Reset (MASR) instruction or direct command to reset the axis.	Shutdown State Error For a motion coordinated instruction, look at the extended error code (EXERR). It identifies which axis caused the error. **Example**: If EXERR is zero, check the axis for dimension zero.
8	The configured axis type is not correct.	Wrong Axis Type For a motion coordinated instruction, look at the extended error code (EXERR). It identifies which axis caused the error. **Example**: If EXERR is zero, check the axis for dimension zero.
9	The instruction tried to execute in a direction that aggravates the current overtravel condition.	Overtravel Condition
10	The master axis reference is the same as the slave axis reference or the Master Axis is also an axis in the Slave Coordinate System.	Master Axis Conflict
11	At least one axis is not configured to a physical motion module or has not been assigned to a Motion Group.	Axis Not Configured For single axis instructions: the Extended Error code for MAG, MDAC, MAPC, MAM, MAJ, MATC, and MCD is defined as: 1 = Slave axis 2 = Master Axis Note that for MAM, MCD, and MAJ in time driven mode, the axis being moved is a slave axis. For multi-axes instructions: the Extended Error code for MDCC, MCLM, MCCM, and MCCD is defined as: The axis number in the coordinate system where 0 = 1st axis 2 = Master Axis or 3rd Slave Axis
12	Messaging to the servo module failed.	Servo Message Failure
13	The value of at least one operand is out of range.	Value Out Of Range
14	The instruction cannot apply the tuning parameters because of an error in the run tuning instruction.	Tune Process Error
15	The instruction cannot apply the diagnostic parameters because of an error in the run diagnostic test instruction.	Test Process Error

Error	Corrective Action or Cause	Notes
16	Wait until the homing process is done.	Home In Process Error
17	The instruction tried to execute a rotary move on an axis that is not configured for rotary operation.	Axis Mode Not Rotary
18	The axis type is configured as unused.	Axis Type Unused
19	The motion group is not in the synchronized state. This could be caused by a missing or mis-configured servo module.	Group Not Synchronized
20	The axis is in the faulted state.	Axis In Faulted State
21	The group is in the faulted state.	Group In Faulted State
22	Stop the axis before you execute this instruction.	Axis In Motion
23	An instruction attempted an illegal change of dynamics.	Illegal Dynamic Change
24	Take the controller out of test mode.	Illegal AC Mode Op
25	You attempted to execute an instruction that is not correct.	Illegal Instruction
26	The cam array is of an illegal length.	Illegal Cam Length
27	The cam profile array is of an illegal length.	Illegal Cam Profile Length
28	You have an illegal segment type in the cam element.	Illegal Cam Type
29	You have an illegal order of cam elements.	Illegal Cam Order
30	You tried to execute a cam profile while it is being calculated.	Cam Profile Being Calculated
31	The cam profile array you tried to execute is in use.	Cam Profile Being Used
32	The cam profile array you tried to execute has not been calculated.	Cam Profile Not Calculated
33	It attempted to execute an MAH instruction without a position cam in process.	Position Cam Not Enabled
34	A MAH instruction is trying to start while a registration is already running.	Registration in Progress
35	Either the controller or the Output Cam module does not support the specified Output Cam, axis, input or output.	Illegal Execution Target
36	The size of the Output Cam array is not supported or the value of at least one member is out of range: • Output bit less than 0 or greater than 31. • Latch type less than 0 or greater than 3. • Unlatch type less than 0 or greater than 5. • Left or right position is out of cam range and the latch or unlatch type is set to “Position” or “Position and Enable”. • Duration less than or equal to 0 and the unlatch type is set to “Duration” or “Duration and Enable”. • Enable type less than 0 or greater than 3 and the latch or unlatch type is set to “Enable”, “Position and Enable”, or “Duration and Enable”. • Enable bit less than 0 or greater than 31 and the latch or unlatch type is set to “Enable”, “Position and Enable”, or “Duration and Enable”. • Latch type is set to “Inactive” and unlatch type is set to either “Duration” or Duration and Enable”.	Illegal Output Cam
37	Either the size of the Output Compensation array is not supported or the value of one of its members is out of range. The array index associated with errors 36 and 37 are stored in .SEGMENT of the Motion Instruction data type. Only the first of multiple errors are stored. The specific error detected is stored in Extended Error Code. With the ability to dynamically modify the Output Cam table, the Illegal Output Cam error 36 can occur while the MAOC is in-process. In general, the cam elements in which an error was detected are skipped. The following are exceptions, and continues to be processed. • Error 2, Latch Type Invalid. Latch Type defaults to Inactive. • Error 3, Unlatch Type Invalid. Unlatch Type defaults to Inactive. • Error 8, with Unlatch Type of Duration and Enable. Behaves as an Enable Unlatch type.	Illegal Output Compensation

Error	Corrective Action or Cause	Notes
38	The axis data type is illegal. It is incorrect for the operation.	Illegal Axis Data Type For a motion coordinated instruction, look at the extended error code (EXERR). It identifies which axis caused the error. **Example**: If EXERR is zero, check the axis for dimension zero.
39	You have a conflict in your process. Test and Tune cannot be run at the same time.	Process Conflict
40	You are trying to run a MSO or MAH instruction when the drive is locally disabled.	Drive Locally Disabled
41	The Homing configuration is illegal. • Home Speed cannot be zero. • Home Return Speed cannot be zero. • For AXIS_SERVO_DRIVE, you have an absolute homing instruction when the Homing sequence is not immediate.	Illegal Homing Config
42	The MASD or MGSD instruction has timed out because it did not receive the shutdown status bit. Usually a programmatic problem caused when either MASD or MGSD is followed by a reset instruction that is initiated before the shutdown bit has been received by the shutdown instruction.	Shutdown Status Timeout
43	You have tried to activate more motion instructions than the instruction queue can hold.	Coordinate System Queue Full
44	You have drawn a line with three 3 points and no centerpoint via point or plane centerpoint can be determined.	Circular Collinearity Error
45	You have specified one 1 point radius or “r;drawn a line” centerpoint, via point and no centerpoint radius or plane centerpoint, via point can be determined.	Circular Start End Error
46	The programmed centerpoint is not equidistant from start and end point.	Circular R1 R2 Mismatch Error
47	Contact Rockwell Automation Support.	Circular Infinite Solution Error
48	Contact Rockwell Automation Support.	Circular No Solutions Error
49	$\|R\| < 0.01$. R is basically too small to be used in computations.	Circular Small R Error
50	The coordinate system tag is not associated with a motion group.	Coordinate System Not in Group
51	You have set your Termination Type to Actual Position with a value of 0. This value is not supported.	Invalid Actual Tolerance
52	At least one axis is currently undergoing coordinated motion in another coordinate system.	Coordination Motion In Process Error
53	Trying to initiate an MAOC or MDOC on an inhibited axis.	Axis Is Inhibited
54	1. Open the properties for the axis. 2. On the Dynamics tab, enter a value for the Maximum Deceleration.	Zero Max Decel You cannot start motion if the maximum deceleration for the axis is zero.
61	See the extended error code (EXERR) for the instruction.	Connection Conflict
62	Cancel the transform that controls this axis or don’t use this instruction while the transform is active.	Transform In Progress You cannot execute this instruction if the axis is part of an active transform.
63	Cancel the transform that controls this axis or wait until the transform is done moving the axis.	Axis In Transform Motion You cannot execute this instruction if a transform is moving the axis.
64	Use a Cartesian coordinate system.	Ancillary Not Supported You cannot use a non-Cartesian coordinate system with this instruction.

Error	Corrective Action or Cause	Notes
65	The axis moved too far and the controller cannot store the position. To prevent this error, set up soft travel limits that keep the axis within the position range. One way to get more travel is to use the max negative or max positive position as your home position. **Example**  **Important**: This error does not apply to a CIP axis.	Axis Position Overflow The range for position depends on the conversion constant of the axis.  • Maximum positive position = 2,147,483,647 / conversion constant of the axis. • Maximum negative position = -2,147,483,648 / conversion constant of the axis. Suppose you have a conversion constant of 2,097,152 counts/inch. In that case: • Maximum positive position = 2,147,483,647 / 2,097,152 counts/inch = 1023 inches. • Maximum negative position = -2,147,483,648 / 2,097,152 counts/inch = -1023 inches. For a motion coordinated instruction, look at the extended error code (EXERR). It identifies which axis caused the error. ExErr#1: Axis 0 Caused the Error ExErr#2: Axis 1 Caused the Error ExErr#1: Axis 2 Caused the Error
66	Be sure to keep the robot in the arm solution that you configured it in. You can configure the robot in either a left arm or right arm solution.	You are attempting to fold back an articulated independent or dependent two axis robot on itself at the quadrant boundaries.
67	• Change the target positions to values that are within the reach of the robot. • If X2b +X2e isn't zero, stay out of this region: 	Invalid Transform Position You are trying to move to a place the robot cannot reach.
68	Move the joints so that the end of the robot isn't at the origin of the coordinate system.	Transform At Origin You cannot start the transform if the joint angles result in X1 = 0 and X2 = 0.
69	• Check the maximum speed configuration of the joints. • Use target positions that keep the robot from getting fully stretched or folding back on itself at the origin of the coordinate system. • Move in a relatively straight line through positions where X1 = 0 and X2 = 0.	Max Joint Velocity Exceeded The calculated speed is very high. This happens when the robot either: • gets fully stretched. • folds back on itself. • moves away from X1 = 0 and X2 = 0 in a different angle than it approached that position. **Example**: These moves produce this error. 
70	Look for source or target axes that are configured as rotary positioning mode. Change them to linear positioning mode.	Axes In Transform Must Be Linear A transform works only with linear axes.
71	Wait until the transform that you are canceling is completely canceled.	Transform Is Canceling

Error	Corrective Action or Cause	Notes
72	Check the target positions. A calculated joint angle is beyond +/- 360·.	Max Joint Angle Exceeded
73	Check that each MCT instruction in this chain is producing valid positions.	Coord System Chaining Error This MCT instruction is part of a chain of MCT instructions. There is a problem with one of the instructions in the chain.
74	Change the orientation to angles that are within +/- 360·.	Invalid Orientation Angle
75	You cannot use this instruction with a SoftLogix controller.	Instruction Not Supported
76	1. Open the properties for the axis. 2. On the Dynamics tab, enter a value for the maximum deceleration jerk.	Zero Max Decel Jerk You cannot start motion that uses an S-Curve profile if the maximum deceleration jerk for the axis is zero.
77	How many axes are in your coordinate system? • 2 — Use a non-mirror transform direction. • 3 — Use a non-inverse transform direction.	Transform Direction Not Supported 1. You're trying to use the mirror directions with a 3-axis coordinate system and a non-zero base offset (X2b) or effector offset (X2e). 2. Mirror directions are not supported for 2-axis Coordinate Systems. 3. You are attempting to use either a 2 or 3-axis Cartesian target coordinate system with transform directions other than forward and inverse. You can use inverse mirror directions only when both these conditions are true: • You have a 3-axis coordinate system. • The base offset (X2b) and end effector offset (X2e) of the X2 dimension are zero.
78	Not Allowed While Stopping	New check for a secondary instruction overlap on top of an active Stop instruction.
79	Error of Home instruction, if any, active pro0file encountered during internal home completion state.	Invalid Planner State
80	Error of MAOC instruction when the Output Connection format is not correct. • Bad Connection Parameter - Connection Instance Failure. Internal error can occur. • Bad Communication Format - I/O subsystem Failure. • CIP Sync not synchronized - Scheduled output module reporting not synchronized to a CIP Sync master. • Grandmaster Clock mismatch - Scheduled output module has different Grandmaster clock than the controller.	Incorrect Output Connection
81	Error on MGSR, if a MASD or MGS (programmed) is executed while the MGSR is still in process.	Partial Group Shutdown Reset.
82	The axis was found to be in the incorrect operational axis state.	CIP axis is incorrect state
83	The MDS instruction cannot be performed due to control mode selection.	Illegal Control Mode or Method
84	The CIP drive device digital input is not assigned.	Drive Digital Input Not Assigned
85	Homing not allowed when a redefine position is in process. Performing MAH while MRP is in process results in this instruction error.	Redefine Position in Process
86	Current use of the MDS instruction requires an optional attribute that is not supported.	Optional Attribute Not Supported
87	The instruction is invalid while running planned motion.	Not Allowed While Planner Active
93	A move was programmed in MDSC mode before the MDSC link has been established by the execution of a MDAC or MDCC.	MDSC Not Activated
94	• Some dynamics units belong to Master Driven Mode and some to Time Driven Mode. • Some units are time based whereas others are velocity based, for example, Speed in Seconds and Acceleration in units/sec2. • Incompatibility of units. Dynamics in Seconds are incompatible with Merge Speed = Current.	MDSC Units Conflict
95	• All instructions in the queue must use a compatible Lock Direction, for example, Position Forward Only and Immediate Forward Only. • Lock Direction = None and speed units belong to Master Driven Mode.	MDSC Lock Direction Conflict
96	MDAC(All) and MDAC(something other than All) on the same slave.	MDSC MDAC All Conflict

Error	Corrective Action or Cause	Notes
97	Trying to replace a running Master with a new Master whose speed is zero, or replacing a Slave that is moving via an MAM with another MAM with the same or a different Master that is not moving.	MDSC Idle Master and Slave Moving
98	The actual direction of master axis' motion does not match the direction programmed by Lock Direction parameter (IMMEDIATE FORWARD ONLY or IMMEDIATE REVERSE ONLY) when the slave is already moving.	MDSC Lock Direction Master Direction Mismatch
99	Parameter combination not supported.	Feature Not Supported • Performing MDCC on non-Cartesian coordinate system • Using Lock Position for MATC in Time Driven Mode
100	If speed is in seconds or Master units, move must start from rest.	Axis Not At Rest
101	Return data array is either nonexistent or not big enough to store all the requested data.	MDSC Calculated Data Size Error
102	Attempt to activate a second MDSC instruction with a Lock Position or a Merge with a Lock Position while an axis is moving.	MDSC Lock While Moving
103	If the Master Axis is changed and the new slave speed is less than 5% of the original slave speed for Single Axis instructions, or 10%, depending on the move of the original Slave Coordinate System speed, then this error occurs and the change is not allowed. **Note:** The same applies when changing from Time Driven mode to MDSC mode.	MDSC Invalid Slave Speed Reduction
104	IF: a motion instruction performs either: • A change in the Master Axis • A change in speed units AND: if in the same update period, the instruction is either forced to pause with a speed of zero, or stopped with a MAS or MCS THEN: the velocity profile is changed to trapezoidal and this error code is reported.	MDSC 2 Instructions were started in 1 Update Period, therefore Jerk was Maximized.
105	An instruction in the coordinated motion queue is either trying to change the Master Axis or changing the mode from MDSC mode to Time Driven mode or from Time Driven mode to MDSC mode.	MDSC Invalid Mode or Master Change
106	You cannot use merge to current when dynamics is programmed in seconds.	Merge To Current Using Seconds Illegal

You get an error if certain Motion Instructions overlap while Motion Stop Instructions are active. In this case, an instruction is actively stopping and a second instruction is initiated that overlaps the active instruction. The table below lists some of the overlap instances that generate errors.

In this case:

- Error # 7 = Shutdown State Error.
- Error #61, ExErr #10 = Connection Conflict, Transform Axes Moving or Locked By Other Operation.
- Error #78 = Not Allowed While Stopping.

Table 76 - Active Stopping Instruction

	MGS			MGSD	MCS			MAS	
Initiated Second Instruction	Stop Mode = Fast Stop	Stop Mode = Fast Disable	Stop Mode = Programmed		Stop Type = Coordinated Move	Stop Type = Coordinated Transform	Stop Type = All	All Stop Types Except Stop Type = All	Stop Type = All
MAAT	Error #78	Error #78	Error #78	Error # 7	Error #78	Error #78	Error #78	Error #78	Error #78
MRAT	Error #78	Error #78	Error #78	Error # 7	Error #78	Error #78	Error #78	Error #78	Error #78

Table 76 - Active Stopping Instruction

MAHD	Error #78	Error #78	Error #78	Error # 7	Error #78	Error #78	Error #78	Error #78	Error #78
MRHD	Error #78	Error #78	Error #78	Error # 7	Error #78	Error #78	Error #78	Error #78	Error #78
MAH	Error #78	Error #78	Error #78	Error # 7	Error #78	Error #78	Error #78	Error #78	Error #78
MAJ	Error #78	Error #78	Error #78	Error # 7			Error #78		Error #78
MAM	Error #78	Error #78	Error #78	Error # 7			Error #78		Error #78
MAG	Error #78	Error #78	Error #78	Error # 7			Error #78		Error #78
MCD	Error #78	Error #78	Error #78	Error # 7	Error #78	Error #78	Error #78	Error #78	Error #78
MAPC	Error #78	Error #78	Error #78	Error # 7			Error #78		Error #78
MATC	Error #78	Error #78	Error #78	Error # 7			Error #78		Error #78
MDO	Error #78	Error #78	Error #78	Error # 7			Error #78		Error #78
MCT	Error #78	Error #78	Error #78	Error # 7	Error #61 ExErr #10	Error #61 ExErr #10	Error #61 ExErr #10	Error #61 ExErr #10	Error #61 ExErr #10
MCCD	Error #78	Error #78	Error #78	Error # 7			Error #78		Error #78
MCLM/MCCM (Merge = Disabled)	Error #78	Error #78	Error #78	Error # 7	Error #78	Error #78	Error #78		Error #78
MCLM/MCCM (Merge=Enabled)	Error #78	Error #78	Error #78	Error # 7		Error #78	Error #78		Error #78

		Active Stopping Instruction						
		MGS			**MGSD**	**MCS**	**MAS**	**MASD**
Initiated Second Instruction	Stop Type	Stop Mode = Fast Stop	Stop Mode = Fast Disable	Stop Mode = Programmed	None	Stop Type = All	Stop Type = All	None
MGS	Stop Mode = Fast Stop	Error #78	Error #78	Error #78	Error #7			
	Stop Mode = Fast Disable	Error #78	Error #78	Error #78	Error #7			
	Stop Mode = Programmed	Error #78	Error #78	Error #78	Error #7			
MGSR	None	Error #78	Error #78	Error #78	Error #7			Error #7
MCS	Stop Type = Coordinated Move	Error #78	Error #78	Error #78	Error #7	Error #78	Error #78	
	Stop Type = Coordinated Transform	Error #78	Error #78	Error #78	Error #7	Error #78	Error #78	
	All Stop Types Except Stopgap = All	Error #78	Error #78	Error #78	Error #7			
MAS	Stop Type != All	Error #78	Error #78	Error #78	Error #7	Error #78	Error #78	Error #7
	Stop Type != All	Error #78	Error #78	Error #78	Error #7			Error #7
MASR	None	Error #78	Error #78	Error #78	Error #7			Error #7

Additional Error Code Information

Refer to your drive user manual for more information about error codes displayed on drives and/or multi-axis motion control systems. Refer to Additional Resources on page 13.

Notes:

History of Changes

MOTION-UM002C-EN-P, September 2012

Change
Where to Find Sample Projects
Reference Position
Transform Position
Data Flow When a Move is Executed with an MCTP - Forward Transform
Data Flow When a Move is Executed with an MCTP - Inverse Transform
Error Code 41
Error Code 50

MOTION-UM002B-EN-P, November 2011

Change
Added External Access and Constant tag parameters
Added Motion Planner Tab
Updated Termination Type Table
Updated MCLM, MCCM, and MCCD Instruction Operand Descriptions
Updated Runtime Error Conditions for Instructions
Updated Coordinate Motion Status Bits table for MCLM and MCCM Instructions
Added Information on Dwells
Added Information on Zero Length Moves
Added Information on Time-based Programming Errors
Updated Three-dimensional Arcs Example
Added Information on Master Driven Speed Control (MDSC) and Motion Direct Command Support
Updated Figure: MCLM Ladder Instruction with Move Type Incremental
Updated Example: MCCM Ladder Instruction for 3D Arc Using Circle Type of Via
Added Information on Master Driven Speed Control (MDSC) and the MCS Instruction
Added Information on Master Driven Speed Control (MDSC) and the MCSD Instruction
Added Extended Error Codes for MCTP Instruction
Added Master Driven Coordinate Control (MDCC) Instruction
Added Status Bits for Motion Instructions (MCLM, MCCM) When MCDD Is Active

Change
Added Changing Between Master Driven and Time Driven Modes for Coordinated Motion Instructions
Added Input and Output Parameters Structure for Coordinate System Motion Instructions
Added Speed, Acceleration, Deceleration, and Jerk Enumerations for Coordinated Motion
Updated Error Codes for Motion Instructions

MOTION-UM002A-EN-P, January 2010

Initial Release.

A

Arm Solution
definition of
configuring 83
Articulated Dependent
base offsets 100
define configuration parameters 99
end effector offsets 101
establish the reference frame 93
establish the reference frame alternate methods 96
identify the work envelope 98
link lengths 99
Articulated Independent
base offsets 63
configuration parameters 61
end effector offsets 64
establish reference frame 55, 61
establish reference frame methods 57
identify the work envelope 59
link lengths 61
axis
inhibit 49, 55

C

Cartesian Gantry
configuration parameters 104
establish reference frame 103
identify the work envelope 104
Cartesian H-bot
configuration parameters 107
establish reference frame 106
identify the work envelope 106
Changing Between Master Driven and Time Driven Modes for Coordinated Motion Instructions 238
Changing the Master Axis 239
Collinear Moves
velocity profiles
termination types 43
Common Action Table for Master Axis 233
Configure 65, 67
Coordinate System Attributes
Accel Status 255
Actual Pos Tolerance Status 255
Actual Position 255
Actual Position Tolerance 256
Axes Inhibited Status 256
Axes Servo On Status 256
Axes Shutdown Status 256
Axis Configuration Faulted 256
Axis Fault 256
Axis Inhibit Status 256
Command Pos Tolerance Status 256
Command Position Tolerance 256
Config Fault 257
Coordinate Motion Status 257
Coordinate System Auto Tag Update 257
Coordinate System Status 257
Decel Status 257
Dynamics Configuration Bits 258
Max Pending Moves 258
Maximum Acceleration 258
Maximum Deceleration 258
Maximum Speed 258
Module Fault 258
Modules Faulted 258
Motion Status 258
Move Pending Queue Full Status 258
Move Pending Status 259
Move Status 259
Move Transition Status 259
Physical Axes Faulted 259
Physical Axis Fault 259
Ready Status 259
Shutdown Status 259
Stopping Status 259
Transform Source Status 259
Transform Target Status 259
Coordinate System Dialog Boxes
Dynamics 19
General 19
Geometry 19
Manual Adjust 19
Offset 19
Tag 19
Units 19
Coordinate System Properties
Dynamics Tab 28
Manual Adjust 30
Reset Button 30
Manual Adjust Button 30
Position Tolerance Box 29
Actual 29
Command 29
Vector Box 28
Maximum Acceleration 28, 31
Maximum Acceleration Jerk 29
Maximum Deceleration 28
Maximum Deceleration Jerk 29
Maximum Speed 28
Editing 20
General Tab 21
Axis Grid 21
Axis Name 22
Brackets 21
Coordinate 21
Coordination Mode 22
Dimension 21
Ellipsis 22
Ellipsis button 21
Enable Coordinate System Auto Tag Update 22
Motion Group 21
New Group button 21
Transform Dimension 21
Type 21
Geometry Tab 23
Link Lengths 23
zero angle orientations box 24
Joints Tab 27
Joint Ratio 27
Joint Units 27
Offsets Tab 26
Base Offsets 26

Tag Tab 32
Data Type 32
Description 32
Name 32
Scope 32
Tag Type 32
Units Tab
Axis Grid 25
Axis Name 25
Conversion Ratio 25
Conversion Ratio Units 25
Coordination Units 24
Coordinate system properties
Offsets Tab
End Effector 26
Coordinated Motion Status Bit 238

D

Delta 64
Delta Robot
Maximum Negative Joint Limit Condition 71
Maximum Positive Joint Limit Condition 70, 71, 73, 74, 75, 76, 78, 80
types
configure 64
Delta three-dimensional
configuration parameters 71
configure 65
maximum positive joint limit condition 70
reference frame 66
work envelope 69
zero angle orientation 67
Delta two-dimensional
configuration parameters 76
configure 73
establish the reference frame 74
work envelope 75

E

End Effector Offsets
determining 100
error
motion instructions 261, 269
error codes
drives 267
motion instructions 261, 269
errors
additional information 267
Establish 66

F

Fault Conditions for Motion Instructions when MDCC Is Active 233

G

Geometry
of robot 52
Geometry Tab
link lengths 23
zero angle orientations box 24

I

Identify 69
inhibit
axis 49, 55
Input and Output Parameters Structure for Coordinate System Motion Instructions 240

K

Kinematics
activating 84
arm solutions 83, 85
arm solutions for two axes robots 83
Articulated Independent 55
changing arm solutions 85
determine Coordinate system type 52
no solution 86
singularity 85
solution mirroring 83
terms 50
kinematics
See multi-axis coordinated motion instructions

L

Logix Designer application 11

M

Maximum Acceleration 28, 31
MCCD
Examples
Impact of Changes to Acceleration and Deceleration Values on Motion Profile 191
Relay Ladder 194
Operands
Relay Ladder 186
Structured Text 189
MCCM
Examples
Circular Error 178
CIRCULAR_COLLINEARITY_ERROR (44) 179
CIRCULAR_R1_R2_MISMATCH_ERROR (46) 180
CIRCULAR_SMALL_R_ERROR (49) 181, 182
CIRCULAR_START_END_ERROR (45) 179
Rotary Axes 161

Move Type of Absolute 161
Move Type of Incremental 163
Structured Text 183
Three Dimensional Arcs 166
Two Dimensional Arc 146
Using Center Circle Type 146
Using Center Incremental Circle Type 156
Using Radius Circle Type 153
Using Via Circle Type 150
Two Dimensional Full Circle 158

MCLM
Examples
Additional Note On Merging Instructions 129
Blending
Different Speeds 48
Merge 126
Rotary Axes
Move Type of Absolute 119
Move Type of Incremental 120

MCS
Examples
Relay Ladder 201
Operands
Relay Ladder 195

MCSD
Examples
Relay Ladder 205

MCSR
Examples
Relay Ladder 229
Structured Text 229

MCT 205

MCT instruction 205

MCTP 49, 217

MDCC 229

motion
error codes 261, 269

Motion Attributes
Motion Coordinate System
Status Attributes
Coordinate Motion Status 255
Motion Coordinate System Configuration Attributes
Coordinate System Dynamics Configuration
Actual Position Tolerance 255
Servo Gains
Acceleration Feedforward Gain 255

Motion Axis Jog 205

motion coordinated instructions
See multi-axis coordinated motion instructions

Motion Coordinated Transform 205

Motion Direct Command and the MDCC Instruction 231

motion instructions
error codes 261, 269

Motion Move Instructions
Motion Axis Jog (MAJ) 205
Description 209
Operands 205
Structured Text 206
Motion Calculate Transform Position (MCTP)
Description 220
Extended Error Codes 222
MOTION_INSTRUCTION data type 219
Operands 217
Structured Text 218

MOTION_INSTRUCTION Bit Leg Definitions for MDCC 232

Multi Axis Coordinated Motion
Circular Programming Reference Guide 183

Multi-Axis Coordinated Motion Instructions 109
Introduction 15
Master Driven Coordinate Control (MDCC) 229
Operands 230
Master Driven Coordinated Control (MDCC)
Arithmetic Status Flags 232
Master Reference 231
Operands
Relay Ladder 230
Structured Text 231
MCCD 185
MCCM 136
MCLM 109
MCS 194
MCSD 202
MCSR 49, 227
MCT 205
MDCC 229
Motion Calculate Transform Position (MCTP) 217
Motion Coordinated Change Dynamics (MCCD)
Arithmetic Status Flags 192
Changes to Status Bits 193
Description 185
Error Codes 192
Extended Error Codes 192
Fault Conditions 192
Operands
Change Speed 191
Motion Control 190
Motion Type 190
Relay Ladder 186
Structured Text 189
Motion Coordinated Circular Move 136
Motion Coordinated Circular Move (MCCM)
Arithmetic Status Flags 176
Changes to Status Bits 183
Axis Status Bits 183
Coordinate System Status Bits 183
Description 136
Extended Error Codes 177
Fault Conditions 176
Operands 138
Circle Type
Center 139
Center Incremental 139
Radius 139
Via 139
Merge 172

All Motion 172
Coordinated Motion 172
Merge Disabled 172
Motion Control 144
Move Type 144
Absolute 145
Incremental 145
Relay Ladder 138
Structured Text 141
Termination Type 41, 43, 45
Actual Tolerance 41, 43, 45
Command Tolerance 41, 43, 45
Follow Contour Velocity Constrained 41, 43, 45
Follow Contour Velocity Unconstrained 41, 43, 45
No Decel 41, 43, 45
No Settle 41, 43, 45
Via/Center/Radius 145
Runtime Error Conditions 177
Target Position Entry Dialog 174
Motion Coordinated Linear Move (MCLM)
Arithmetic Status Flags 132
Changes to Status Bits 134
Axis Status Bits 135
Coordinate Motion Status Bits 135
Coordinate System Status Bits 135
Description 110
Extended Error Codes 133
Fault Conditions 133
Operands
Accel Jerk 125
Dwells 130, 172
Jerk Units 125
Merge 126
All Motion 126
Coordinated Motion 126
Merge Disabled 126
Motion Control 115
Move Type 116
Absolute 116
Incremental 116
Profile 122
S-Curve 123
Trapezoidal 123
Velocity Profile Effects 123
Relay Ladder 110
Structured Text 113, 229
Termination Type 41, 43, 45
Actual Tolerance 41, 43, 45
Command Tolerance 41, 43, 45
Follow Contour Velocity Constrained 41, 43, 45
Follow Contour Velocity Unconstrained 41, 43, 45
No Decel 41
No Settle 41, 43, 45
Time Based Programming Errors 130, 173
Zero Length Move 130
Runtime Error Conditions 133
Symmetric Profiles 45
Target Position Entry Dialog 131
Velocity Profiles 43
Motion Coordinated Shutdown (MCSD)
Arithmetic Status Flags 203
Changes to Status Bits 204
Axis Status Bits 204
Coordinate Motion Status Bits 205
Coordinate System Status Bits 204
Description 202
Error Codes 204
Fault Conditions 203
Operands 202
Motion Control 203
Relay Ladder 202
Structured Text 203
Motion Coordinated Shutdown Reset 227
Motion Coordinated Shutdown Reset (MCSR)
Arithmetic Status Flags 228
Changes to Status Bits 228
Axis Status Bits 228
Coordinate Motion Status Bits 229
Coordinate System Status Bits 228
Description 49, 227
Error Codes 228
Fault Conditions 228
Operands 227
Motion Control 228
Relay Ladder 227
Structured Text 227
Motion Coordinated Stop (MCS) 194
Arithmetic Status Flags 200
Changes to Status Bits 201
Axis Status Bits 201
Description 194
Extended Error Codes 201
Fault Conditions 201
Operands 194
Decel Rate 200
Error Codes 201
Motion Control 196
Relay Ladder 195
Structured Text 196
Motion Coordinated Transform (MCT) 205

N

Naming a Coordinate System 16

R

RSLogix 5000 software Verification Errors 234

S

SCARA Delta
configuration parameters 80
establish the reference frame 78
identify the work envelope 80
SCARA Independent
reference frame 86, 88
Selective Compliant Assembly Robot Arm
configuration parameters 89
configure 83
establish reference frame 86
identify work envelope 88
link lengths 89
Singularity
planning for
definition of 85
Speed, Acceleration, Deceleration, and Jerk Enumerations for Coordinated Motion 250
Acceleration and Deceleration Enumerations 250
Jerk Enumerations 252
Speed Enumerations 250
Status Bits for Motion Instructions (MCLM, MCCM) when MDCC Is Active 234
Studio 5000 Engineering and Design Environment 11
Symmetric Profiles
paths of 45

T

transform
start a transform 205
troubleshoot
drive errors 267
instruction errors 261, 269

V

Velocity Profiles
of collinear moves 43
triangular 47

Notes:

Rockwell Automation Support

Rockwell Automation provides technical information on the Web to assist you in using its products. At http://www.rockwellautomation.com/support you can find technical and application notes, sample code, and links to software service packs. You can also visit our Support Center at https://rockwellautomation.custhelp.com/ for software updates, support chats and forums, technical information, FAQs, and to sign up for product notification updates.

In addition, we offer multiple support programs for installation, configuration, and troubleshooting. For more information, contact your local distributor or Rockwell Automation representative, or visit http://www.rockwellautomation.com/services/online-phone.

Installation Assistance

If you experience a problem within the first 24 hours of installation, review the information that is contained in this manual. You can contact Customer Support for initial help in getting your product up and running.

United States or Canada	1.440.646.3434
Outside United States or Canada	Use the Worldwide Locator at http://www.rockwellautomation.com/rockwellautomation/support/overview.page, or contact your local Rockwell Automation representative.

New Product Satisfaction Return

Rockwell Automation tests all of its products to help ensure that they are fully operational when shipped from the manufacturing facility. However, if your product is not functioning and needs to be returned, follow these procedures.

United States	Contact your distributor. You must provide a Customer Support case number (call the phone number above to obtain one) to your distributor to complete the return process.
Outside United States	Please contact your local Rockwell Automation representative for the return procedure.

Documentation Feedback

Your comments will help us serve your documentation needs better. If you have any suggestions on how to improve this document, complete this form, publication RA-DU002, available at http://www.rockwellautomation.com/literature/.

Rockwell Automation maintains current product environmental information on its website at http://www.rockwellautomation.com/rockwellautomation/about-us/sustainability-ethics/product-environmental-compliance.page.

Rockwell Otomasyon Ticaret A.Ş., Kar Plaza İş Merkezi E Blok Kat:6 34752 İçerenköy, İstanbul, Tel: +90 (216) 5698400

www.rockwellautomation.com

Power, Control and Information Solutions Headquarters

Americas: Rockwell Automation, 1201 South Second Street, Milwaukee, WI 53204-2496 USA, Tel: (1) 414.382.2000, Fax: (1) 414.382.4444
Europe/Middle East/Africa: Rockwell Automation NV, Pegasus Park, De Kleetlaan 12a, 1831 Diegem, Belgium, Tel: (32) 2 663 0600, Fax: (32) 2 663 0640
Asia Pacific: Rockwell Automation, Level 14, Core F, Cyberport 3, 100 Cyberport Road, Hong Kong, Tel: (852) 2887 4788, Fax: (852) 2508 1846

Publication MOTION-UM002D-EN-P - February 2015
Supersedes Publication MOTION-UM002C-EN-P - September 2012

Copyright © 2015 Rockwell Automation, Inc. All rights reserved. Printed in the U.S.A.

Allen-Bradley

Motion Coordinate System

User Manual